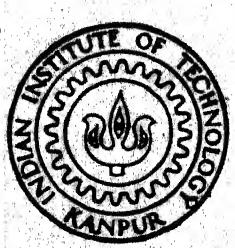
COMPUTING THE SKEPTICAL CLOSURE OF INHERITANCE NETS

by

SWARUP KUMAR MOHALIK



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY, 1992

Title of the Thesis

COMPUTING THE SKEPTICAL CLOSURE OF INHERITANCE NETS

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

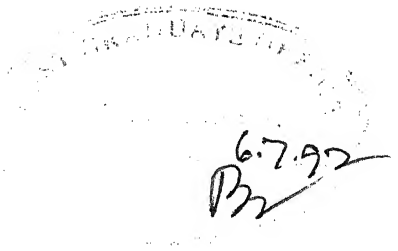
by

Swarup Kumar Mohalik

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGG.
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

July, 1992



CERTIFICATE

Certified that the work contained in the thesis entitled "*Computing the Skeptical Closure of Inheritance Nets*", by Swarup Kumar Mohalik, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Harish Karnick
Computer Science and Engg.
I.I.T. Kanpur.

July, 1992

ACKNOWLEDGEMENTS

My heart-felt thanks to Dr. Karnick for his confidence me and for his guidance and encouragement throughout the thesis. I will forever remember his encouragement for independent thinking, and of course, his lessons in good writing.

The long stay at IITK was made possible by Dr. S. P. Mohanty, in whose association, I never felt like away from home. I will miss the Sunday lunches at his place, and the series of jokes after that.

The unforgettables

Shreesh, the greatest yet the strangest of them all, who is still an enigma to me. The nice face of the thesis is all due him. Pawanji with a sense of humour (and a bottlefull of glucon-C), few people have access to. Chanchal, Alok, Tomator(Nagarkoti), Neeraj, Rahul, Sudhir(Batuk Bhai), SSG, Vineet, Anil Dash, Peeyush(pandey and sonkar), Bosty(Anurag Srivastava), Hemant, the wing named C-TOP where I spent the best two years of life, and the volley court that witnessed the mirth of the crazy C-Toppers.

Ashi, with whom I shared some jokes, some pains. Deepak Gupta, who introduced me to LaTeX. Vermaji, whose pocket and room I explored and exploited at will. Bapat and the ever smiling Ghati Junta.

Ashok bhai, his reassuring presence and his affection. Sarat Das, Sounak, Panda, Deba bhai, Dandapat Sir and his escapades, all the Oriya gang and the countless feasts.

All the Hall-4ites who, some or other time, tolerated my intolerable phattas but still shared my smile.

ABSTRACT

Inheritance networks represent class-subclass relations or simple membership relations through hierarchically structured links between nodes which denote individuals or classes. When the class-subclass relations are defeasible, we have to represent the exceptions as well. Allowing multiple inheritance with exceptions in the nets lead to conflicting proofs in the net. Such conflicts are resolved by allowing the more specific proof to override the less specific proof. The most used specificity relation in the literature says that a subclass is more specific than a superclass. This entails a preemption strategy in the nets where less specific proofs are preempted by more specific proofs. However, a problem arises when specificity can not resolve the conflict. Choosing one proof as against the other leads to different sets of inferences, called extensions. Some inferences can, however, be drawn in all the extensions, although they may have different proofs in different extensions. This set of inferences is called the skeptical closure of the inheritance net.

We adopt an alternative resolution mechanism, where if the preemption strategy fails, we choose any and only one of the contradictory proofs from some pre-specified preference set. This simple mechanism gives us a procedure for computation of skeptical inheritance without recourse to the more obvious method of generating all extensions and then taking the intersection.

Since inheritance nets represent only one aspect of knowledge in the world, it should be possible to express inference in nets in general frameworks for commonsense reasoning. Autoepistemic logic(AEL) is one such framework which models the beliefs of an ideally rational agent. To adopt this logic for our purpose, we first translate the nets into the TMS(Truth Maintenance Systems) formalism with suitable modifications in order to incorporate the preference set information. We call this modified formalism PTMS(Parametrized TMS). Since TMS has direct correspondence with AEL, the semantics of inheritance computation can be given in terms of AEL. The translation to PTMS generates a large number of justifications. Hence, we give an alternative formalism, called ITMS(Inheritance TMS), which exploits the regularity of information in the nets and generates fewer number of justifications. Then we show the equivalence between PTMS-translation and ITMS-translation of inheritance nets.

This thesis first deals with inheritance networks where all the inheritance relations are assumed to be defeasible and where the net is acyclic. The case of cyclic nets with defeasible links is tackled next. We suggest a modification of the specificity relation and a modified procedure for computation of skeptical inheritance for these nets. Skeptical inheritance in the most general nets where cycles may be present and where links may represent both strict and defeasible relations is dealt with last. This is the most general result so far in the literature.

Contents

1 Introduction

1.1 Inheritance Systems

An inheritance system is a representation system founded on the hierarchical structuring of knowledge. Here, the regularity present in the knowledge is exploited by viewing the individual entities at different levels of detail and defining abstractions by a collection of properties common to each level. For example, a knowledge base may have reference to a number of elephants named, say, Clyde, Jumbo and Ernie. The knowledge about these elephants may be represented by associating with each a separate table describing the properties of each elephant, that it is *gray, four-legged, one-trunked,* and *one-tailed* animal, and that its name is so and so. But since this will lead to duplication, an alternative representation which is compact and conceptually simpler is creating an abstraction called *Elephant* having the common properties of all the elephants and mentioning each of Clyde, Jumbo and Ernie as instances of this abstraction. (See figure 1.1).

Abstractions can also share properties just as individuals do. So we may also define new abstractions *over abstractions.* For example, see figure 1.2. The two abstractions *Man* and *Elephant* have some properties in common: they are warm-blooded and they suckle their young. We may define an abstraction called *Mammal* over them denoting this set of properties and make Man and Elephant it's instances.

This type of structuring knowledge through abstractions leads to class-subclass hierarchies. They have several advantages.

The primary advantage of a hierarchical system is that it is an efficient method of knowledge representation. The abstractions considerably reduce duplication of information, hence updating is far easier.

The other advantages are

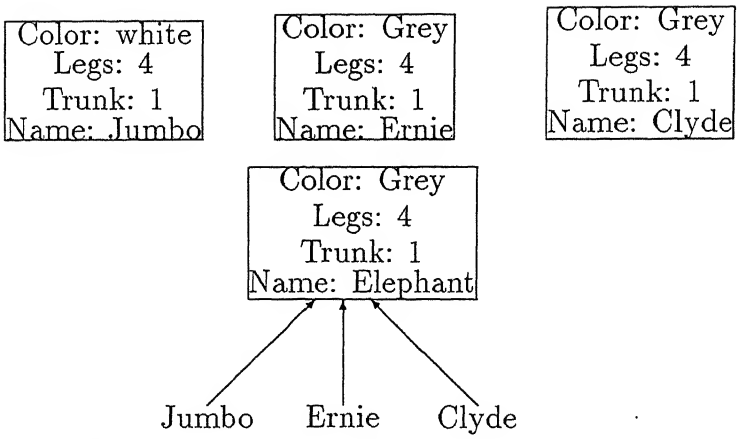


Figure 1.1: Abstraction of properties.

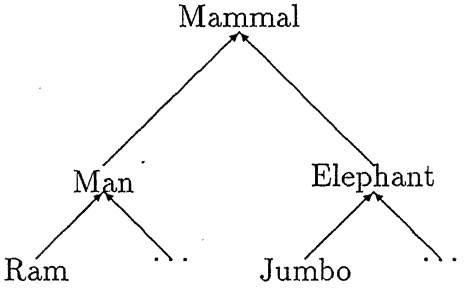


Figure 1.2: Abstraction over abstractions.

- Searching is more efficient in the organized hierarchy than in an unorganized list of explicit assertions about individuals.
- it enables representing real world knowledge as we conceptually visualize it.
- Most important, it permits reasoning in the presence of incomplete information.

The following example from [13] illustrates the above points.

Suppose we have to determine the chemical properties of a chemical compound. One naive way is to perform individual tests to determine each chemical property separately. But this approach may prove to be infeasible if the sample amount is small or if it is expensive. A better approach is to perform a smaller number of tests and classify the compound as being an acid or a base, organic or inorganic etc. One can then draw conclusions about the properties of the compound using the properties associated with the class of acids, bases, organic or inorganic compounds since the compound is a member of the class. Thus such hierarchical organization allows us to fill in gaps in our knowledge where the required knowledge is impossible or expensive to acquire.

In a hierarchical system, if a property P is associated with a class C, and an individual

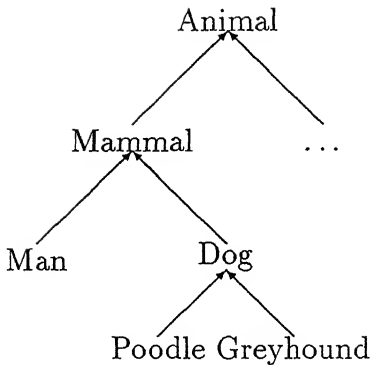


Figure 1.3: A taxonomic hierarchy.

i belongs to C, then we may say i *inherits* P from C. The knowledge representation systems employing the hierarchical organization of knowledge are therefore called *Inheritance Systems.* The pictorial representation of such systems is, in general, a graph; we call this an *Inheritance Network, Inheritance Net, Inheritance Hierarchy, Inheritance graph,* or just a *net.* The membership or the class/subclass relations are referred to as *inheritance relations.*

1.1.1 Taxonomic Hierarchies

An inheritance hierarchy where the abstractions are organized as a tree (see figure 1.3) is known as a *Taxonomic Hierarchy.* Here an individual or class inherits *directly* all the properties from its unique parent, i.e. an individual or a class possesses all the properties associated with itself and with all its ancestors. For example, if Fido is a *Poodle*, then by inheritance it possesses all the properties of *Dog* and *Mammal.* This is implicit in the representation.

1.1.2 Multiple inheritance

When representing the knowledge that an individual may inherit properties from several different and orthogonal abstractions, taxonomic hierarchies lead to unnecessary duplication since every individual/class is restricted to at most one parent node. For instance, see figure 1.4.

To represent several elephants who are circus performers we have to create a subclass of circus performers and name it *Circus performer elephant* and restate all the properties of *Elephant* for that class.(Equivalently, we can creat a subclass of elephants called *Elephants that are circus performers* and restate the properties of circus performers for it.) Either

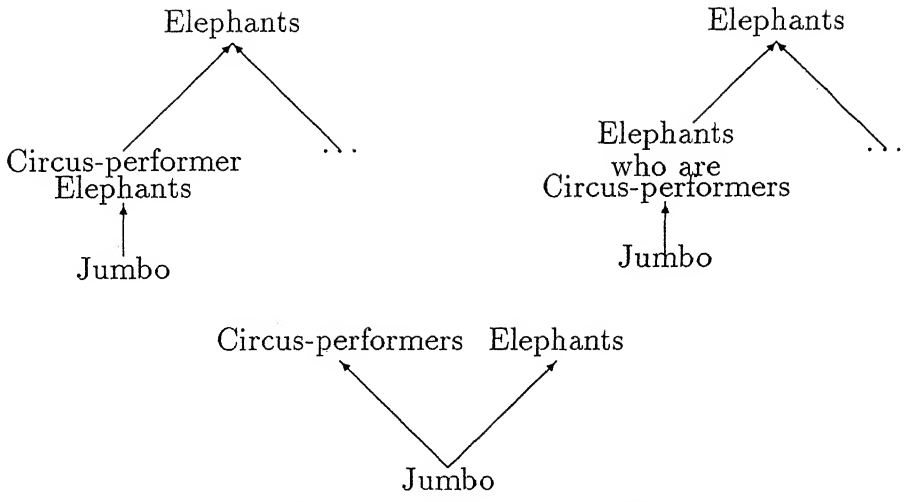


Figure 1.4: Multiple inheritance.

way, there is duplication of information. Further, the choice between *Elephant* and *Circus performer* for the major class is arbitrary, hence undesirable.

Allowing an individual/class to inherit from *multiple* superclasses solves this problem of duplication. Here the inheritance tree is replaced by a *directed* inheritance graph. Thus an elephant Clyde an inherit simultaneously from the classes *Elephant* and *Circus performer*. The individual/class inherits all the properties associated with itself and all its ancestors.

1.1.3 Inheritance Nets with Exceptions

An abstraction over a group of individuals is based on a set of common properties that the individuals share. An individual inherits a majority of properties associated with the class. But it is common in the real-world to have exceptional members that do not possess some properties that the majority of the individuals in the class possess. As a matter of fact, any realistic representation of real-world knowledge must necessarily allow for representation of exceptions. For example, if we have an abstraction of *Elephant* as *grey, quadruped* animals and also that Clyde is a while elephant, then Clyde inherits from *Elephant* all th properties except greyness.

Classes may also have exceptional properties. For example, see figure 1.5. Birds generally share the *Flying* property; but Penguins are exceptions to this. *Flojo* is a gifted penguin that can fly, so she is an exception to the class of normal penguins and so on. This kind of abstraction with the possibility of exceptions necessitates a new relation that specifies a normal subset relation between classes and superclasses, except for the exceptions of the class. We call this a *defeasible* relation.

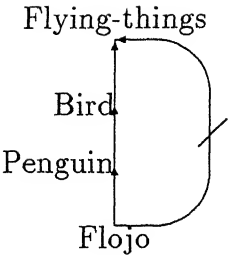


Figure 1.5: Exceptions in a net.

1.1.4 Multiple Inheritance with Exceptions

While taxonomic hierarchies with exceptions and hierarchies with multiple inheritance(no exceptions) have been amenable to solution, multiple inheritance together with exceptions gives rise to several problems. The bulk of the literature is an attempt to understand and solve the problems related to such nets. In this thesis, we deal with the problem of inference in nets where multiple inheritance with exceptions is implicitly assumed.

1.2 The problem at Hand

1.2.1 Knowledge Representation in The Nets

The information specified in the network can be classified into the following three categories [13].

- *Explicit knowledge*: These are the explicitly stated relations among individuals and classes. The relations basically reflect inheritance information and are of the following types.

    1. *Strict relations :* They denote either the membership of an individual in a class or a subset relation among classes and superclasses. The assertions *Tweety is a bird, Ram is a man, All men are mortal, All dogs are mammals* etc. are examples of strict relations.

    2. *Defeasible relations* or Defaults : They denote a subset relation among classes and superclasses where exceptions are allowed and allow us to make reasonable conjectures in the absence of complete information. *Birds normally fly, Quakers are normally pacifists* etc. are examples of such defaults. The meaning of such normative relations has been a moot point in AI literature [11, 12, 15]. The meaning we attach to the defaults is that in the absence of information to

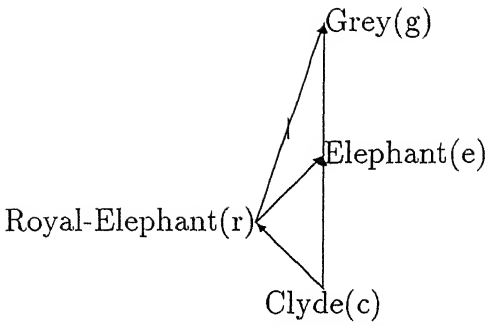


Figure 1.6: Preemption of proofs.

the contrary the defaults behave as strict relations. For example, if along with the above examples, we have just that *Tweety is a bird*, then we may deduce that *Tweety flies*. But if we are also given that *Tweety has broken wings and hence can not fly* the default about birds is no more applicable.

- *Implicit knowledge*: These are facts that are deduced from the explicit knowledge in the network. For example, if we have the explicit knowledge that *Birds fly, All men are mortal, Ram is a man* and *Tweety is a bird*, the implicit knowledge that can be deduced from it is that *Ram is mortal* and *Tweety flies*.

Ideally, inheritance nets should be able to deduce the implicit knowledge when both types of relations stated above are present.

1.2.2 Conflicts in The Net and Their Resolution

In order to deduce implicit knowledge, or in other words, to *draw inferences* in the nets, we form chains of reasoning; we call them *proofs* for the *inferences*. To simplify things now and later, whenever we talk of inferences, we assume that they are about a particular object, an individual or a class, which we call a *focus*. This does not result in any loss of generality, since we may make different objects the focus and draw inferences about them. Let us consider the following example(see figure 1.6).

Example 1.1 The statements we have are:

Elephants are grey.
Royal elephants are not grey.
Royal elephants are elephants.
Clyde is a royal elephant.

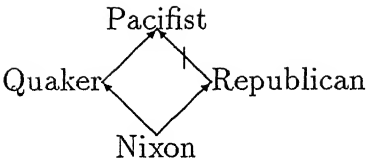


Figure 1.7: Unresolved conflicts.

We may form an proof like this:*Since Clyde is a royal elephant, and royal elephants are not grey, Clyde is not grey.* So given that Clyde is the focus, we may draw the inference that it is non-grey through this proof.

The presence of exceptions in the nets leads to conflicting proofs, that is, proofs leading to contradictory inferences. One has to resolve the conflicts to get an unequivocal inference.

In the previous example, along with the proof for the non-greyness of Clyde, we also have the following proof: *Since Clyde is a royal elephant, royal elephants are elephants, and elephants are grey, Clyde is grey.* These two proofs conflict over the greyness of Clyde. But intuitively we feel that Clyde is not grey, even though he is an elephant, because he is a special type of elephant: a royal elephant. This intuition that information about subclasses should get precedence over information about superclasses forms a strategy for resolving conflicting proofs in nets. We call this the *preemption strategy* where proofs based on less specific information(of superclasses) are preempted by proofs based on more specific information(of subclasses).

There may be other strategies for resolving conflicts among proofs. The preemption strategy is, by far, the most accepted.

Problem arises when the resolution strategies fail to resolve a conflict between two proofs. Consider the following example(see figure 1.7).

Example 1.2 The statements are:

Quakers are pacifists.
Republicans are not pacifists.
Nixon is a Quaker.
Nixon is a republican.

The preemption strategy can not resolve between the two conflicting proofs for the inference *whether Nixon is a pacifist or not*, because Quaker is neither a subclass nor a superclass of Republican. We say the inference is *ambiguous.*

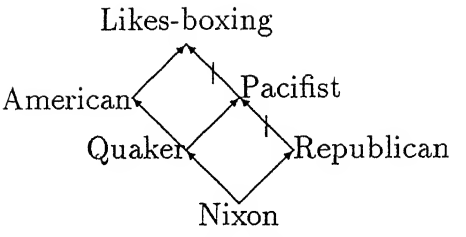


Figure 1.8: Credulous extensions.

The only way such ambiguities can be resolved is by *choosing* one proof as against the other. Because of the arbitrariness of the choice, we get different sets of inferences, each set being independent of the other. They are called *credulous extensions* of the net[18]. While the notion of ambiguity and credulous extensions is well-studied, the idea of resolving an ambiguity through a choice is new and interesting, because we are then able to characterize each credulous extension by a set of choices.

Example 1.3 See figure 1.8. The focus node is Nixon. The inference pacifist(with respect to the focus) is ambiguous, because there are two unresolvable proofs: Nixon through quaker to pacifist and Nixon through republican to non-pacifist. One may choose that Nixon is a pacifist or a non-pacifist. Further, if Nixon is a pacifist, then likes-boxing is ambiguous because of the unresolvable proofs: Nixon through quaker through American through likes-boxing and Nixon through quaker through pacifist through not-likes-boxing. One may again choose among these proofs. If Nixon is a non-pacifist, however, this ambiguity does not arise and we may safely draw the inference that Nixon likes-boxing. Thus, the extensions are the following: first, where pacifist and likes-boxing is chosen, second where pacifist and not-likes-boxing is chosen and third where non-pacifist is chosen. The sets of inferences are, respectively, {quaker, republican, pacifist, American, likes-boxing}, {quaker, republican, pacifist, American, not-likes-boxing} and {quaker, republican, non-pacifist, American, likes-boxing}. The credulous extensions are shown in the figure in order.

Each credulous extension represents one possible state of the world. The inferences that hold in every possible state of the world, or in each credulous extension, can be computed by taking the intersection of the inferences of every such extension. We call this intersection the *skeptical closure* of the net. The computation of skeptical inheritance in a net is the computation of its skeptical closure.

Example 1.4 This example is from [17]. See figure 1.9. When *seedless grape vine* is the focus, the inference *fruit plant* is ambiguous. If we choose the *seedless grape vine* to be a

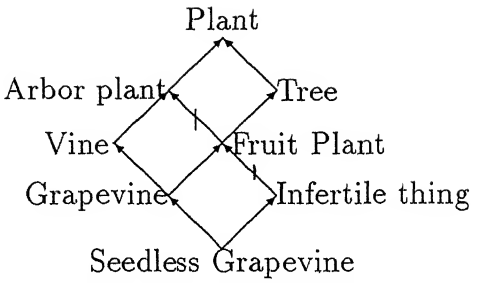


Figure 1.9: Skeptical inheritance.

fruit plant, then *arbor plant* is ambiguous. So we get the credulous extensions: { seedless grape vine, grape vine, infertile thing, fruit plant, vine, arbor plant, tree, plant}, {seedless grape vine, grape vine, infertile thing, not-a fruit plant, vine, arbor plant, plant} and { seedless grape vine, grape vine, infertile thing, fruit plant, vine, not-an arbor plant, tree, plant}. The credulous extensions are shown in the figure in order. The skeptical closure of the net consists of the following inferences: { seedless grape vine, grape vine, infertile thing, vine and plant}.

1.2.3 Inheritance in a general framework

The advent of nonmonotonic logics have been used for formalizing and mechanizing commonsense reasoning. Since the normative statements in inheritance nets capture only a subset of the relations in the real-world, the problem of inheritance can be seen as a subproblem of general commonsense reasoning. Thus, one can expect that an inheritance reasoner can be embedded in a general nonmonotonic reasoning system. This will then provide inheritance with a semantics defined in terms of a nonmonotonic logic.

1.2.4 Statement of The Problem

The problem we tackle in this thesis is to compute skeptical inheritance in nets where all types of relations described above are present and where the resolution strategies are through preemption and choice. The methodology should also project inheritance as a subproblem of some general nonmonotonic logic.

1.3 Outline of The Thesis

Chapter 2 describes the basic notation for representing knowledge in inheritance nets. The problems related to the resolution strategies are explored. This brings into focus the inadequecies of existing systems and the need for alternative approaches. The chapter also discusses some tools needed for the solution of the problem and the plan of attack.

Chapter 3 formalizes acyclic inheritance nets where only the defeasible relations are represented(existing literature is only concerned with such nets) This also provides a correspondence to a standard nonmonotonic logic and on the other, provides an algorithm for computing skeptical inheritance in the nets. The formalization paves the way to deal with more general nets.

In Chapter 4, we extend the notions of Chapter 2 to cyclic nets, and then to nets with both strict and defeasible relations. The incremental generalizations clearly bring out the reasons for modifying some traditional notions in the context of inheritance systems.

We conclude with a discussion about extensions to the proposed system and future work in Chapter 5.

2 Exploring The Problem

This chapter introduces some basic concepts and notation for representation of knowledge in an inheritance net. The problem of inferencing in such nets and the work done so far is discussed. In the last section, we describe our approach to the problem using some tools from the literature.

2.1 Basic Concepts

2.1.1 Objects, Relations And Links

An *object* is either an individual or a class. Letters from the alphabet refer to individuals or classes. At times, however, we refer to individuals/classes directly by their name.

The link type $x \Rightarrow y$ represents a positive strict relation *All x's are y's*. A negative strict relation *No x's are y's* is represented by the link type $x \not\Rightarrow y$. Since, by contraposition, *No x's are y's* is equivalent to *No y's are x's* ($y \not\Leftarrow x$), we often represent this negative strict relation between x and y by the link type $x \not\Leftrightarrow y$. When x is an individual these strict relations are nothing but the simple membership relation. For example, we may represent the statements *All dogs are mammals* and No man is omniscient by $dog \Rightarrow mammal$ and $man \not\Leftrightarrow omniscient\text{-}beings$ respectively. $Ram \Rightarrow man$ denotes *Ram is a member of the class Man*, while $Tom \not\Leftrightarrow child$ denotes *Tom is not a member of the class child*.

The link type $x \rightarrow y$ represents a positive defeasible relation *x's are (normally) y's* and A negative defeasible relation *x's are (normally) not y's* is represented by the link type $x \not\rightarrow y$. For example, the statement *Birds fly* can be represented by $bird \rightarrow fly$, if fly stands for the class of all flying things. Similarly, the statement *Indians are not*

individualists can be represented by $Indian \not\rightarrow Individualist$.

Often we use statements like *Gopal is normally shy.* But this "normally" refers to the temporal status of Gopal, stating that Gopal is shy most of the time. When we ignore this temporal dependence of membership relation, at a particular time, Gopal is either shy or not shy and there is nothing normative about the membership relation. Hence, such membership relation is always represented as a strict relation. The representation, then, is $Gopal \Rightarrow shy$. If we gather information to the contrary at some point of time, we just negate the membership relation and represent the fact as $Gopal \not\Rightarrow shy$(Gopal is not-shy).

Calligraphic capital letters represent networks(graphs with nodes and link types as described above). Networks are strict if they contain only strict links, defeasible if they have only defeasible links and mixed if they contain both defeasible and strict links .

Pictorially, the defeasible links are shown as thin arrows and the strict links as thick arrows. The negative links have a little "cut" or "slash" on them.

2.1.2 Proofs And Paths

Proofs are represented by certain sequences of links called *paths* in the net. Consider the following example.

Example 2.1 See figure 2.1, where Some sentences and the corresponding links have been shown. The proof *Since birds are flying-things, and flying-things (normally) have wings, birds have wings* for the inference *birds have wings* can be represented by a path birds$\rightarrow$ flying-things$\rightarrow$ have-wings. The proof *yogis are Indians, and Indians are not-individualists* for the inference *yogis are not individualists* is represented by the path $yogis \rightarrow Indians \not\rightarrow individualist$. The presence of the strict relation *All selfish men are individualists* implies we can draw the inference *yogis are not selfish* by the path $yogis \rightarrow Indians \not\rightarrow individualist \Leftarrow selfish$.

The representation of proofs by sequences of links presumes a *transitivity"* of proofs, that is, if we have a sequence $a \rightarrow b \rightarrow c$, then from the link $a \rightarrow b$, we infer $a \rightsquigarrow b$ and from the link $b \rightarrow c$; we infer $b \rightsquigarrow c$, and hence by transitivity, we infer $a \rightsquigarrow c$. This assumption of transitivity is not valid in general, when defeasible links are in the sequence. For example, given the sequence $whale \Rightarrow mammal \rightarrow land\text{-}animal$, transitivity will lead to the conclusion $whale \rightsquigarrow land\text{-}animal$, which is grossly incorrect. But, when we interpret the defeasible links autoepistemically(meaning, a link $a \rightarrow b$ is interpreted as "if a is true and

Birds fly	bird$\rightarrow$ flying-things
Those who fly have wings	flying-things$\rightarrow$ have-wings
Yogis are Indians	yogis$\rightarrow$ Indians
Indians are not Individualists	Indians$\not\rightarrow$ individualists
All Selfish people are individualists	selfish$\Rightarrow$ individualist

Figure 2.1:

if $\neg b$ is not derivable, then b is true"), then it is easy to see that transitivity holds for all sequences. In simple terms the interpretation demands that in case there is an exception to a derived proof, it must be stated. In the example given above, it means that, in order to have the right result, we have to state the exception $whale \not\rightarrow land\text{-}animal$. Since we intend to interpret the nets through autoepistemic logic, the assumption of transitivity is valid, and the representation of proofs by sequence of links is tenable.

The following notation from [6, 17] is used to represent paths in a net. Paths are classified as simple or compound, strict or defeasible, positive or negative. Lower case greek letters σ and τ(at times subscribed) stand for arbitrary paths. The notation $\pi(x.\sigma.y)$ is used to abbreviate an arbitrary positive path from x to y; $\pi(x.\sigma.\neg y)$ stands for an arbitrary negative path from x to y. In both cases, x is called the *head*, and y, the *tail* of the path.

The simple paths are just the direct links(representation of the *given* inheritance relation between objects)- classified as strict or defeasible, positive or negative, according to the link types. The compound paths(having more than one link) are defined as follows:

1. if $\pi(x.\sigma.y)$ is a strict positive path, then $\pi(x.\sigma.y)\Rightarrow z$ is a strict positive path; $\pi(x.\sigma.y)\not\Leftrightarrow z$ is a strict negative path; $\pi(x.\sigma.y)\rightarrow z$ is a positive defeasible path; $\pi(x.\sigma.y)\not\rightarrow z$ is a defeasible negative path.

2. if $\pi(x.\sigma.\neg y)$ is a strict negative path, then $\pi(x.\sigma.\neg y) \Leftarrow z$ a strict negative path.

3. if $\pi(x.\sigma.y)$ is a positive defeasible path, then $\pi(x.\sigma.y)\Rightarrow z$ and $\pi(x.\sigma.y)\rightarrow z$ are positive defeasible paths; $\pi(x.\sigma.y)\not\Leftrightarrow z$ and $\pi(x.\sigma.y)\not\rightarrow z$ are negative defeasible paths.

4. if $\pi(x.\sigma.\neg y)$ is a defeasible negative path, then $\pi(x.\sigma.\neg y) \Leftarrow z$ is a defeasible negative path.

The various type of paths have been shown in figure 2.2.

$a \rightarrow b \rightarrow c \rightarrow d \rightarrow$	positive defeasible path
$a \rightarrow b \not\rightarrow c$	negative defeasible path
$a \Rightarrow b \Rightarrow c$	positive strict path
$a \Rightarrow b \not\Leftrightarrow c$	negative strict path
$a \Rightarrow b \not\Leftrightarrow c \Leftarrow d$	negative strict path
$a \rightarrow b \rightarrow c \Rightarrow d$	positive defeasible path
$a \rightarrow b \rightarrow c \not\Leftrightarrow d$	negative defeasible path
$a \rightarrow b \rightarrow c \not\Leftrightarrow d \Leftarrow$	negative defeasible path

Figure 2.2:

We define the *strict end segment* of a path σ as the maximal strict end segment of σ. For example, if σ is $x \Rightarrow y \rightarrow p \not\Leftrightarrow r \Leftarrow s$, then the strict end segment of σ is $p \not\Leftrightarrow r \Leftarrow s$.

All inferences are about inheritance relations between objects. So when we draw an inference A *inherits from* B, we represent it by $A \rightsquigarrow B$. When we have an inference A *does not inherit from* B, we represent it by $A \not\rightsquigarrow B$.

When we assume a focus A, we may refer to an inference just by B(or $\neg B$), when it will mean an inference $A \rightsquigarrow B$(or, $A \not\rightsquigarrow B$) .

A *valid* inference about the inheritance relationship between objects is drawn on the basis of a *valid* proof. The following machinery carries over the notion of validity from proofs to paths in the net.

Paths, representing proofs, *enable* certain inferences as their conclusions. A positive path $\pi(s.\sigma.x)$ enables the inference $s \rightsquigarrow x$(or just x if the focus is s). A negative path $\pi(s.\sigma.\neg x)$ enables the inference $s \not\rightsquigarrow x$.

Given a network $\mathcal{G}$, the goal is to find the set of valid inferences from the set of statements given in the net. The validity of an inference is defined as below.

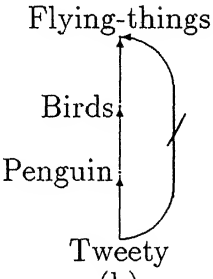
Definition 1 An inference is valid *holds or is supported or is allowed* in a net $\mathcal{G}$ iff it is *enabled* by a path which $\mathcal{G}$ *permits.*

If $\mathcal{G}$ permits a path σ, we call it a valid path and denote it as $\mathcal{G} \triangleright \sigma$.

It remains only to define the *permission* relation. Since this relation is relative to a resolution strategy, we postpone the definition to the next section.

Tweety is a penguin	tweety $\rightarrow$ penguin
All penguins are birds	penguin $\rightarrow$ bird
Birds fly	bird $\rightarrow$ $flying - thing$
Penguins do not fly	penguin $\not\rightarrow$ flying-things

(a)



(b)

Figure 2.3: Tweety's Plight.

2.2 Pinpointing the Problems

In this section, we try to find the specific problems involved in derivation of the implicit knowledge specified in the nets. We do this gradually, starting from *acyclic defeasible nets*. Incidentally, this is the only class of inheritance nets that have been the object of study in the literature. The more general *cyclic defeasible nets* and the most general *mixed nets* need some generalization of the extant notions of specificity. We explore them in order.

2.2.1 Acyclic Nets

The inheritance net is represented as a directed, acyclic graph with objects as nodes and the edges being the defeasible link types. Even the strict relations are represented through defeasible links only.

For example, see figure 2.3(b), which is a graph representation of the defeasible relations given in figure 2.3(a).

2.2.1.1 The permission relation

The resolution strategy of preemption uses the specificity information contained in proofs.

In the following definitions, we assume a focus s.

Definition 2 [Specificity]

A node B is more specific than a node A, written $B <_S A$ if B is on a path $\pi(s.\sigma.B.\tau.A)$ which is permitted by the net $\mathcal{G}$.

Definition 3 [Preemption]

A path $\sigma = \pi(s.\tau_1.A \rightarrow B)$ is preempted in $\mathcal{G}$ iff there is a path $\delta = \pi(s.\tau_2.C \not\rightarrow B)$ such that $C <_S A$ by the path $\pi(s.\tau_2.C.\tau_3.A)$. The preemption of a negative path is similar.

We now give the definition of the permission relation in the acyclic nets. This definition is incomplete, since it does not take into account the issue of ambiguity. But it serves to give an insight into how complex proofs can be built using the preemption strategy.

Definition 4 [Permission]

$\mathcal{G} \triangleright \tau$ iff either

1. $\tau = S \rightarrow B or S \not\rightarrow B$ for some node B or
2. if τ is a compound path $\pi(s.\sigma.x)$ then
   (a) $\mathcal{G} \triangleright \pi(s.\sigma)$, and
   (b) $\forall \delta = \pi(s.\sigma_1.D \ni \mathcal{G} \triangleright \delta$, and $D \not\rightarrow A \in \mathcal{G}$,
   τ is not preempted by $\pi(s.\sigma_1.D \not\rightarrow B)(D \not<_S A)$.

The intuition behind this definition is the following:

- Since the paths $S \rightarrow B$(or, $S \not\rightarrow B$) represent explicit facts about the focus, they should always hold.
- A compound path is permitted, only if all its subpaths starting from the focus node are permitted, which reflects an upward flow of properties.
- A path can be preempted only by a valid path(a path permitted by the net).

The definition is mutually dependent but not circular, since the acyclic net can be ordered topologically and validity of a path from s to x depends only on the nodes strictly earlier in a topological order. The proof is given later for the general definition of permission that takes into account the ambiguities.

Example 2.2 See figure 2.4. Since $a \rightarrow e \rightarrow f$ is permitted since there is no conflict. Hence e is more specific than f. This leads to the preemption of the path $a \rightarrow f \rightarrow g$. Since the path to he only negative child of d is not permitted, the path $\sigma = a \rightarrow b \rightarrow c \rightarrow d$ is permitted.

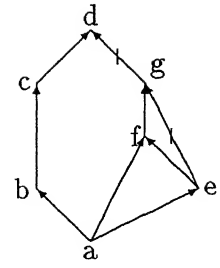


Figure 2.4: Preemption.

The idea of preemption as a resolution strategy dates back to [3, 18]. Touretzky [18] introduced a restricted form of preemption, which is referred to as *on-path preemption.* Here, the more specific node which triggers the preemption is restricted to be on the path which is being preempted. The example in figure 2.3 illustrates on-path preemption: the path $tweety \rightarrow penguin \rightarrow bird \rightarrow flying\text{-}things$ is on-path preempted by the path $tweety \rightarrow penguin \not\rightarrow flying\text{-}things$, since penguin is on the path which is preempted.

Sandewall [16] noted that on-path preemption was too restrictive. Its inadequacy is evident from figure 1.6. Here on-path preemption lets the path $clyde \rightarrow elephant \rightarrow grey$ be permitted by the net; this result is certainly counter-intuitive. Sandewall introduced off-path preemption where the more specific node can be on any other valid path in the net as the node *royal-elephant* is on the path $clyde \rightarrow royal-elephant \rightarrow elephant$ and not on the path $clyde \rightarrow elephant \rightarrow grey$.

The definition we have given is more general and captures both on-path and off-path preemption and is *in vogue* in the current literature.

2.2.1.2 Ambiguity and Extensions

Ambiguity arises when two supported paths in an inheritance net conflict.

Definition 5 Let $\mathcal{G}$ be an inheritance net. A node t in $\mathcal{G}$ is *ambiguous* with respect to a node s, denoted as $s \diamond t$, if there are two paths $\pi(s.\sigma.t)$ and $\pi(s.\sigma.\neg t)$ such that they both are permitted in $\mathcal{G}$.

The net $\mathcal{G}$ is said to ambiguous if there is a node t which is ambiguous with respect to the focus node.

The relativity of ambiguity is necessitated by the fact that the same node may or may not be ambiguous with respect to different nodes in the net.

Example 2.3 See figure 2.5. In the net G, node e is ambiguous with respect to the node a, but is not so with respect to the node b.

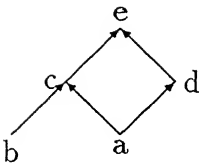


Figure 2.5: Relativity of Ambiguity.

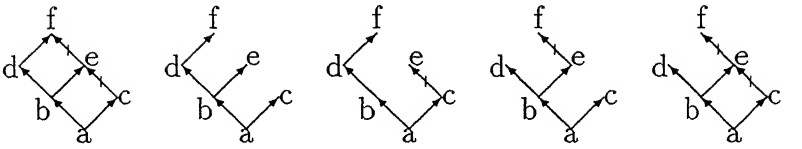


Figure 2.6: Credulous extensions.

An ambiguity means there are equally good reasons for and against the relation between the nodes and that the specificity criterion has not been able to resolve this conflict. So in order to disambiguate this conflict, one must choose one path over the other. This choice leads to credulous extensions.

Definition 6 A credulous extension [19, 17] of an inheritance net $\mathcal{G}$ with respect to a focus node a is a maximal unambiguous subgraph of $\mathcal{G}$. Any superset of the extension is ambiguous with respect to a.

Example 2.4 See figure 2.6. In (a), we have the net $\mathcal{G}$. In (b), (c) and (d) we have the three credulous extensions of $\mathcal{G}$. In each of the extensions, if we add an extra edge, then there is an ambiguity. In (b), adding the edge $c \not\rightarrow e$ results in an ambiguity at e. Adding $e \not\rightarrow f$ to (b) leads to an ambiguity at f. These non-extensions are shown in (e) and (f) respectively.

As we have noted, there may be more than one credulous extension with respect to a focus node. The set of extensions may be different for different foci. One may accept one of these extensions arbitrarily as a state of the world.

Skeptical Inheritance, on the other hand, generates a single set of inferences, which is the intersection of the inferences supported by each of the credulous extensions of the net. The intersection is called the *skeptical closure.* An inference which is in the skeptical closure is called *skeptically acceptable.* Formally, the skeptical closure of a net $\mathcal{G}$ with the focus a, denoted as Σ_a, is defined as:

$\Sigma_a = \{a \rightsquigarrow [\neg]x \mid \forall$ credulous extensions Γ_a of $\mathcal{G}$, $\Gamma_a \triangleright a \rightsquigarrow [\neg]x\}$.

The example 1.4 illustrated the skeptical closure of the net in figure 1.9.

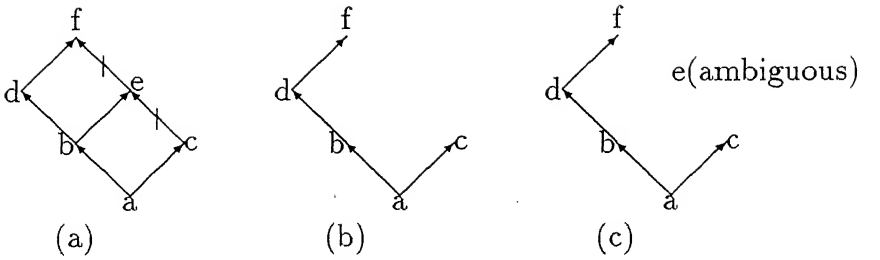


Figure 2.7: Ambiguity Blocking and Propagating Inheritance.

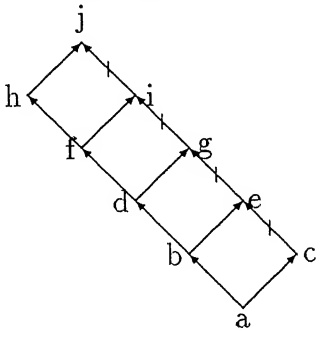


Figure 2.8: Parity Bias in HTT.

2.2.1.3 Problems in Presence of Ambiguity

Computing Skeptical Inheritance

Stein[17] discusses two path-based approaches to compute the skeptical inheritance in a net. In [5], which was the first attempt at a skeptical approach to inheritance nets, that is, to find a unique, unambiguous set of inferences in any inheritance hierarchy, without recourse to finding all the credulous extensions and then taking the intersection. Here, a path rooted at the focus is blocked as soon as an ambiguous node is reached, by deleting the node and the links associated with it from the net. This approach is called the *ambiguity blocking inheritance.*

Example 2.5 See figure 2.7. The focus is a. The node e is ambiguous with respect to a, so it is deleted from the net. Then $a \rightsquigarrow f$ is supported in the net since there in no conflict.

The second approach for computing skeptical inheritance is the ambiguity propagating approach, where an ambiguous node x is marked, but is treated as if the inference $a \rightsquigarrow x$ is supported. This allows an ambiguous line of reasoning to proceed. For example, in figure 2.7.c, the node e is marked as ambiguous first. Since the path $a \rightarrow b \rightarrow e$ is assumed to hold, now the node f also is ambiguous, hence it is also marked as ambiguous.

Now we consider the cases where the above approaches give unintuitive results. This

will then provide an insight into the source of the problem in computing skeptical inheritance. We refer to the blocking approach by HTT(Horty, Thomason and Touretzky) and the propagating approach by ST(Stein).

Zombie Paths: This example is due Makinson et al. [9]. Consider figure 2.7. According to HTT, the path $a \rightarrow b \rightarrow e$ is blocked because $a \diamond e$. Hence, the inference $a \rightsquigarrow f$ is always true, because there is no path that contradicts $a \rightarrow b \rightarrow d \rightarrow e$. But certainly this is not the correct conclusion, because the inference $a \rightsquigarrow e$ may still hold some times, and at those times the inference $a \not\rightsquigarrow f$ may be a possibility.

Parity Bias: This example is from [17]. Consider the figure 2.8. According to HTT, $a \diamond e, a \rightsquigarrow g, a \diamond h, a \rightsquigarrow j, \ldots$. Further, $a \rightsquigarrow j, b \diamond j, d \rightsquigarrow j, f \diamond j, \ldots$. This shows HTT computes a kind of *parity* on the number of ambiguities in the net and there is no intuitive reason why the conclusions thus reached should be correct. Such parity bias is actually a consequence of the blocking mechanism of HTT which has been criticised above.

Floating Conclusion: The example is originally from [17] but the name of "floating conclusion" is due [9]. An inference α is called "floating" if every extension supports α but there is no path common to all the extensions that supports α . Consider the figure 2.9. It is the old "grapevine" example renamed. According to ST, we have $a \diamond e, a \diamond f$ and $a \diamond h$. But actually it is not so. Let us analyze the problem. It is true that $a \diamond e$. If "a is an e", then "a is an h", because of the uncontested path $a \rightarrow b \rightarrow e \rightarrow g \rightarrow h$. If "$a$ is not an e", then e is blocked. Hence the path $a \rightarrow b \rightarrow d \rightarrow f \rightarrow h$ is permitted. So no matter how we resolve the ambiguities(equivalently, no matter which extension we consider), we have that "a is an h". The inference $a \rightsquigarrow h$ is true always.(It is worth noting that the paths $a \rightarrow b \rightarrow e \rightarrow g \rightarrow h$ and $a \rightarrow b \rightarrow d \rightarrow f \rightarrow h$ are by themselves ambiguous since e and f are ambiguous.)

conditional preemption: See figure 2.10. According to HTT, e is deleted since $a \diamond e$. Hence $a \not\rightsquigarrow i$ is supported in HTT. But the result is incorrect. Let us see why. Certainly, $a \diamond e$. If "a is not an e", then e is effectively blocked. Hence the path $a \rightarrow d \rightarrow h \not\rightarrow i$ is permitted in the net and $a \not\rightsquigarrow i$ holds. If "a is an e", then $a \diamond h$. If "a is not an h", then h is blocked, hence $a \rightarrow b \rightarrow e \rightarrow i$ is permitted. If, on the other hand, "a is an h", it is through the path $a \rightarrow b \rightarrow e \rightarrow g \rightarrow h$, so e is more specific than h. That means the paths $a \rightarrow b \rightarrow e \rightarrow g \rightarrow h \not\rightarrow i$ and $a \rightarrow d \rightarrow h \not\rightarrow i$ are preempted by the path $\sigma = a \rightarrow b \rightarrow e \rightarrow i$. σ itself is not preempted by any path, so it is permitted and the inference $a \rightsquigarrow i$ holds. According to ST, $a \diamond e$, hence $a \diamond h$, and hence $a \diamond i$ is inferred. But this does not give us any information about the nature of inferences that results from a particular choice at the ambiguities(i.

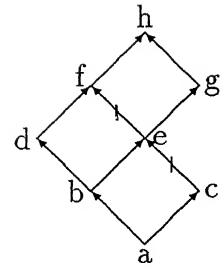


Figure 2.9: Floating Conclusions.

e. inferences in different extensions). There is an insufficiency of information derived in ST. One can see easily that this is the reason for which ST was unable to derive the right conclusion in case of the above example of floating conclusions.

The above examples show the inadequacy of HTT and ST to capture the right intuition behind the notion of ambiguities. In the presence of an ambiguity, HTT is too strict in disallowing the possibility of the node participating in proofs in some state of the world; ST is too loose in saying that an ambiguous inference makes further proofs ambiguous, thus discarding the possibility that while individual chains of reasoning may be ambiguous, collectively they may conclude unambiguously about an inference(as in the case of floating conclusions).

We discard both these approaches in favour of a new approach which maintains that an ambiguity at a is resolved only by *choosing* one of the two conflicting inferences. This choice is arbitrary but once the choice is made about an inference, say a, we are constrained to the extensions in which the inference a holds. This immediately suggests a simple method to represent extensions: A constraint set(we later call it a preference set) {[¬]a [¬]b}, [¬]c ...} stands for the sets of extensions where the inferences [¬]a, [¬]b, [¬]c, etc. are whenever there is a conflict present at the respective nodes.

With this notion of constraint sets, we define the full permission relation as follows. This definition is for a defeasible positive path. Definition for a defeasible path is similar.

Definition 7 [Permission]

$\mathcal{G} \mid\!\!\rhd \tau$ iff either

1. $\tau = S \rightarrow B$ or $S \not\rightarrow B$ for some node B , or
2. if τ is a compound path $\pi(S.\sigma.A \rightarrow B)$ then
   (a) $\mathcal{G} \mid\!\!\rhd \pi(S.\sigma.A)$, and

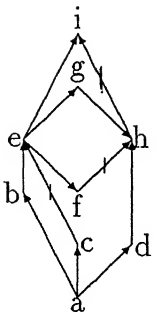


Figure 2.10: Conditional Preemption.

(b) there is no negative path that preempts the path under consideration and there is no conflict, i. e. $\forall \delta = \pi(s.\sigma_1.D) \ni \mathcal{G} \rhd \delta$, and $D \not\rightarrow A \in \mathcal{G}$,
τ is not preempted by $\pi(s.\sigma_1.D \not\rightarrow B)(D \not<_S A)$ and $\pi(s.\sigma_1.D \not\rightarrow B)$ is preempted by $\tau(A \not<_S)$.

(c) if there is a conflict at B then "B" is chosen , i. e. if $\pi(s.\sigma_1.D \not\rightarrow B)$ is not preempted by $\tau(A \not<_S D)$ and vice versa then "B" is in the constraint set chosen.

The definition of permission relation is inductive. In order to prove that this definition is not circular, we need to associate a measure of complexity to the paths in the net such that we can decide whether $\mathcal{G} \rhd \sigma$ if we know whether $\mathcal{G} \rhd \sigma'$ for each σ' less complex than σ. For this purpose, we introduce the concept of *degree* of paths in the nets.

Definition 8 The degree of a path σ in the net $\mathcal{G}$ - written $deg_{\mathcal{G}}(\sigma)$- is defined as the longest path in $\mathcal{G}$ from the initial node of σ to its end node where no edge is repeated.

Note that if $deg_{\mathcal{G}}(\sigma) = 1$ then σ is a direct path $S \rightarrow B$ or $S \not\rightarrow B$, where S is the focus. Also that the definition of degree is well defined in both acyclic and cyclic nets. When we identify nodes with proofs (paths from the focus to the node), we can define the degree of a node as follows.

Definition 9 The degree of a node A in the net $\mathcal{G}$ - written $deg_{\mathcal{G}}(A)$- is defined as the degree of a path from the focus to the node.

For example, see figure 2.10. The degree of the nodes e, h and i are 2, 4 and 5 respectively.

When the permission relation is not circular, the check for permission of a path in the net terminates. We say that the permission relation is *well-defined* in this case.

Theorem 1 The definition of permission relation given above is well-defined in acyclic nets.

Proof :

Consider any acyclic net $\mathcal{G}$. Then, if there is a path from A to B , then $deg_{\mathcal{G}}(A) < deg_{\mathcal{G}}(B)$. We prove the theorem by induction on the degree of paths.

When $deg_{\mathcal{G}}(A) = 1$, there is a direct link from the focus to the node A is the focus. This simple path is permitted by definition of permission.

For any other path, say $\alpha = \pi(s.\sigma.A \rightarrow B)$, the permission relation is defined in terms of subpaths $\pi(s.\sigma.A)$ and the paths $\pi(s.\tau.C.\tau_1.A)$. But all these paths have degrees strictly less than $deg_{\mathcal{G}}(A)$. Hence, circularity is not possible.

□

2.2.1.4 Remarks

Since the constraint sets effectively characterize extensions, associating these constraint sets with a node is an effective way of expressing the extensions in which the node holds. Then in order to compute the skeptical closure of a net, one just has to have a mechanism to infer when these constraint sets range over all the extensions of the net.

Stein[17] has proposed an algorithm for computing skeptical inheritance in an acyclic defeasible net. The idea is to encode the choices under which an inference $a \rightsquigarrow x$ holds(Stein calls the encoding the label of x, denoted by [x]) by some propositional formulae built from the choices. Then, if this formula is w, the inference $a \rightsquigarrow x$ holds iff w is true. If w is true, then $a \rightsquigarrow x$ always holds , that is, it is in the skeptical closure. The encoding of choices at nodes called *labeling* is done as follows: Label[s] = true, if s is the focus node. For any other node p, Label[p]= $B \wedge (\overline{A} \vee p)$, where $B = \bigvee Label(p^+{}_i)$ and B= $\bigvee Label(p^-{}_i)$. The nodes p^+ are the positive children of p and p^- are the negative children of p. In figure 2.11, we reproduce an example of labeling given in [17]. It labels the net shown in the figure 2.9

Though the work of Stein is restricted to on-path preemption, it provides an insight into the basic idea of encoding extensions by inferences. We extend this later to compute skeptical inheritance in more general nets.

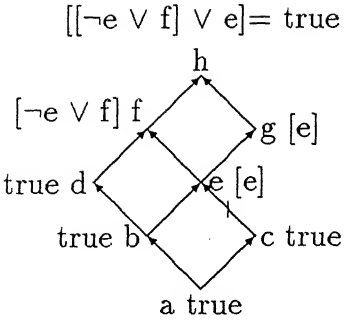


Figure 2.11: Labelling of nodes by Stein.

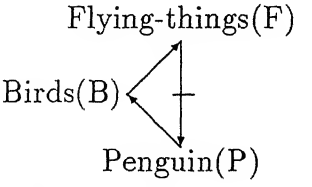


Figure 2.12: Failure of chaining.

2.2.2 Introducing Cycles

Almost all of the extant literature on inheritance is restricted to acyclic networks, since this allows us to have a partial ordering on the specificity relation among the nodes of the net and hence to resolve conflicting arguments by preferring arguments based on more specific information. As Stein[17] has noted, the permission relation is well-defined only for acyclic nets, because we can sort the nodes topologically. This is not so for cyclic nets. The following examples illustrate the difficulty.

Example 2.6 In figure 2.12, the link *flying-things*$\not\rightarrow$*penguin* is natural enough. But when we have the focus *p*enguin, with the present notion of "chaining", we end up with *a penguin is not a penguin*, which is absurd.

Example 2.7 See figure 2.13. Since the permission of each of the inferences $a \rightsquigarrow b$ and $a \rightsquigarrow c$ requires the inference of the other, the permission relation described above can not infer anything about b or b. But, since there are arguments both for and against the nodes, one of the arguments must hold in a state of affairs. Also because of the circularity, the inferences $a \rightsquigarrow b$ and $a \rightsquigarrow c$ become mutually dependent. If $a \rightsquigarrow b$ then $a \diamond c$, because of the paths $a \rightarrow g \rightarrow b$ and $a \rightarrow i \rightarrow c \rightarrow f \not\rightarrow b$. Similarly, if $a \rightsquigarrow c$ then $a \diamond b$. On the other hand, if $a \not\rightsquigarrow c$, then $a \rightsquigarrow b$ since the path $a \rightarrow i \rightarrow c \rightarrow f \not\rightarrow b$ is now blocked at c. Similarly, if $a \not\rightsquigarrow b$ then $a \rightsquigarrow c$. This interdependence of inferences necessitates the characterization of extensions by assuming one of these inferences. Note that these assumptions are different from the assumptions made at ambiguous nodes in the sense that now one has to make a choice among different nodes. This type of choice also leads to credulous extensions.

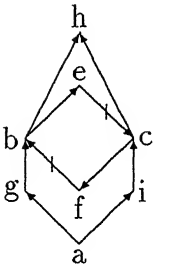


Figure 2.13: Interdependence of Ambiguities.

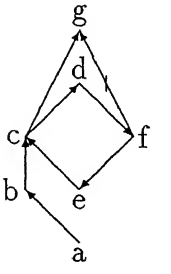


Figure 2.14: Dependency.

Computation of skeptical inheritance must take care of this problem. In the figure 2.13, h is in the skeptical closure of the net.(In the extension where $a \rightsquigarrow b$ is permitted, $a \rightsquigarrow h$ is also permitted through the path $a \rightarrow g \rightarrow b \rightarrow h$ and in the extension where $a \not\rightsquigarrow b$ is permitted, $a \rightsquigarrow c$ is permitted, hence $a \rightsquigarrow h$ is permitted through the path $a \rightarrow i \rightarrow c \rightarrow h$).

Example 2.8 In figure 2.14, there is no way in which we can infer whether a is or is not g, since the paths $a \rightarrow b \rightarrow c \rightarrow g$ and $a \rightarrow b \rightarrow c \rightarrow d \rightarrow f \rightarrow g$ preempt each other according to the present definition of preemption(because c is more specific to f and vice versa). But this seems counterintuitive. The proof of $a \rightsquigarrow c$ is used to derive a proof of $a \rightsquigarrow f$ at the first place. Hence, it is absurd to use the proof of $a \rightsquigarrow f$ to find the proof of $a \rightsquigarrow c$ again(following the cyclic chain), since the proof of $a \rightsquigarrow f$ itself is predicated upon the proof of $a \rightsquigarrow c$. One should allow, in this case, that c is more specific than f.

Example 2.9 Figure 2.15 shows an example similar to the previous one. But, here, there is a difference. We have a proof of $a \rightsquigarrow b$ which does not use the proof of $a \rightsquigarrow c$ and vice versa(note the paths $a \rightarrow g \rightarrow b$ and $a \rightarrow i \rightarrow c$). Because of the independent proofs, we have that "b is more specific to c"(through the path $a \rightarrow g \rightarrow b \rightarrow e \rightarrow c$) and that "c is more specific to b"(through the path $a \rightarrow i \rightarrow c \rightarrow f \rightarrow b$). There is no partial ordering of specificity!! We can not resolve the ambiguity at h.

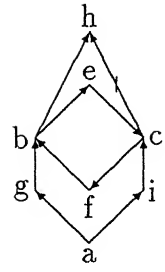


Figure 2.15: Pseudo dependency.

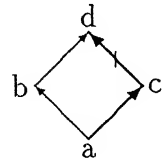


Figure 2.16: Priority of Strict Information.

2.2.2.1 Remarks

The examples suggest a revision of the existing definitions for building proofs through "chaining" and of "specificity". The second example where one has to capture the inter-dependence of ambiguities is harder and is dealt with much later.

2.2.3 Mixed Nets

Any sufficiently rich commonsense domain has to involve a mixture of strict and defeasible information. When such information is represented in an inheritance net, new problems arise[5], as strict and defeasible information are mixed.

We define the following.

Definition 10 [Strict Consequence] The *strict consequences* $\mathcal{S}$ of a path $\delta = \pi(a.\sigma.x)$ in a net $\mathcal{G}$ is defined as: $\mathcal{G}_\delta = \{x \rightsquigarrow y \mid$ there is a strict path from x to y in $\mathcal{G}\ \}$

2.2.3.1 Priority of Strict Information

If we have conflict between a strict path and a defeasible path, the inference enabled by the strict path is to be allowed always, because of the necessity of strict information. In figure 2.16, the inference $a \rightsquigarrow [\neg]d$ is allowed, in spite of the presence of a conflicting path $a \rightarrow b \rightarrow d$.

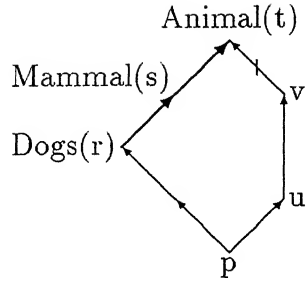


Figure 2.17: Indirect Conflicts.

2.2.3.2 Introduction of cycles

The same figure 2.16 shows a cyclic negative defeasible path $a \rightarrow b \rightarrow d \not\Leftrightarrow c \Rightarrow a$(because $a \rightarrow b \rightarrow d \not\Leftrightarrow c$ is a negative defeasible path, say, σ; so $\sigma \Leftarrow a$ is a negative defeasible path) in the seemingly acyclic net. The introduction of cycles may lead to the same problems as mentioned before.

2.2.3.3 Indirect Conflicts

In defeasible nets, the conflicts involve paths that have identical heads. In the presence of strict links, however, less direct conflicts are possible i. e. the conflicting paths need not have identical heads. See figure 2.17. The strict segment $r \Rightarrow s \Rightarrow t$ means if any object happens to be an r then it must also be a t. Because of this necessity, if the inference $p \rightsquigarrow r$ is allowed, all the inferences $p \rightsquigarrow w$, such that there is a strict path from r to w, are also allowed. Hence conflict at the node r may arise if any of the paths for the inferences $p \rightsquigarrow w$ conflict with another path in the net. In the figure, there are conflicting paths $pqrst$ and $puvt$, hence we can conclude that the paths pqr and $puvt$ conflict. These paths do not share end nodes.

2.2.3.4 New preemption Relation

In figure 2.18(which is sufficiently self-explanatory), according to the observation above, the paths $a \Rightarrow s \rightarrow r$ and $a \Rightarrow s \Rightarrow q \rightarrow p$ conflict. But as one can observe, because of the strict link $r \Rightarrow p$, we have a direct evidence about the native speakers of Pennsylvanian Dutch that they are normally born in America, while we do not have a direct evidence to the contrary. It is only for a superclass "native speakers of Germany" of "native speakers of Pennsylvanian Dutch" that we have evidence to the contrary. Hence the path $a \Rightarrow s \rightarrow r$ actually preempts the path $a \Rightarrow s \Rightarrow q \rightarrow p$. But this can not be captured without modifying the present definition of preemption.

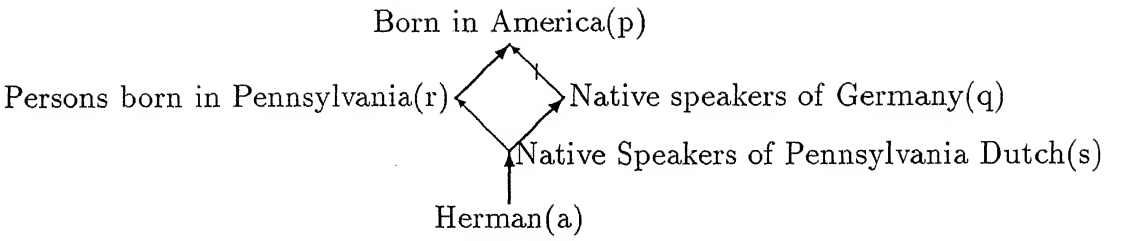


Figure 2.18: New Relations of Preemption.

2.2.3.5 Remarks

Since all the above problems are related to the strict consequences, the definition of preemption has to incorporate this while resolving conflicts between paths.

2.3 Work Done So Far

The development of specially-tailored path-based approaches to solve the problem of inheritance started with [18], as an attempt to purge the system NETL[18] of apparently unintuitive results. Touretzky[18] introduced the notion of *inferential distance ordering* which is the precursor of *on-path preemption*. The system used both *downward* and *upward* reasoning to compute the valid paths in the net. Downward reasoning assumes properties flow from classes to subclasses while the flow is opposite in case of upward reasoning. Sandewall[16] introduced off-path preemption.

The primary advantage of such path-based reasoning is that it is able to exploit the deep underlying structure of the nets. The description is direct and hence appeals to intuition. The disadvantages are that all these systems seem like a complex algorithm for computation of inheritance, the semantics of which is hard to grasp. When strict links are added, the inheritance algorithm becomes even more complicated[6]. For cyclic nets, no path-based solution has been proposed yet.

Even in the case of acyclic defeasible nets, the direct computation of skeptical inheritance by these path-based approaches has not been successful. One has to compute all the credulous extensions and take the intersection of the inferences allowed in each extension.

Translating the inheritance nets into some standard formalism has its advantages over path-based approaches.

1. The primary advantage is that we immediately get the semantics of nets through the semantics of the formalism.

2. Since the presence of normal links leads to a translation into a standard nonmonotonic logic, the idea of the domain of inheritance as a part of commonsense knowledge can be projected through the translation.

3. The inferences in the nets will now be through the sentences of the formalism. Hence they will be uniform in spite of the presence of both normal and strict links, and perhaps with the presence of cycles.

The approaches of [3, 10, 4] are translation based. The disadvantage with this approach is that such a translation may not be simple. The encoding of the natural hierarchical information of nets into the formalism may turn out to be difficult and even faulty. Treating the defeasible links as normal defaults [3] does not capture the specificity information. Hence [3] introduced semi-normal defaults. For complex nets, however, it is very difficult to specify such defaults. Konolige[7] tries to integrate the hierarchical information in Autoepistemic logic(a standard logic for commonsense reasoning) by introducing a HAEL(Hierarchic autoepistemic logic). But the main shortcoming here is that it does not take care of ambiguity and has left the problem dangling.

Prasad in [13] has proposed an evidence-based semantics of inheritance networks where the evidences of class/subclass relations are represented by associating "priority constants" with the nodes. These are used to disambiguate the conflicts. But the problem is how to compute the priority constants for the relations to convert arbitrary nets into evidential logic. Prasad mentions:

> "... to translate an arbitrary inheritance network into evidential logic, the specificity relations implicit in the topology of the network has to be determined first..."

But this is just begging the question.

2.4 Our Approach

We summarize the requirements for a complete solution for computation of inheritance in nets.

1. The solution must be able to handle the general definition of specificity; for cyclic nets perhaps the extant definition has to undergo a change.

2. There must be a way to encode the conditions(choices - or as we call it later, preferences) under which an inference holds; also the mechanism should be able to infer when an inference holds for all the conditions(in [17], it was simple). This seems a minimal requirement for computing skeptical inheritance if one does not want to compute all credulous extensions.

3. Above all, the system must not just specify some complex algorithm, but must have a semantics matching our intuition.

The central idea behind the proposed solution is a translation of the nets to a truth maintenance system(TMS)[2]. The following section explains informally what a TMS is.

2.4.1 Truth Maintenance System

Truth Maintenance Systems(TMS) attempt to provide an effective computational framework for nonmonotonic reasoning. The idea here is to record dependencies among data needed to achieve some result(antecedents) and the computed datum(consequent). For example, if we want to infer that c follows from a and b , then we may inform the TMS about this by introducing a justification

$$c \leftarrow a, b$$

. Then if we want to change some basic decision or assumption, then the data that depends on it can be isolated. Thus revision of current information with the change in assumptions becomes easier.

Nonmonotonicity arises because the inferences we draw not only depend upon the presence of information in our database, but also on the absence of information. Supporting such reasoning requires nonmonotonic justifications:

$$c \leftarrow (a)(b)$$

This means that c depends on the presence of a and the same time on the absence of b. (a) is called **IN**-list and (b) is called **OUT**-list. For example, to capture *Birds normally fly*, we can have the justification

$$fly(x) \leftarrow (bird(x))(\neg fly(x))$$

which says if x is a bird and it can not be derived in the database that x does not fly, then x indeed flies. A TMS maintains, at one time, a consistent set of data known to the system(the current set of beliefs).

The TMS we will use is a propositional TMS, that is, the antecedents and the consequent are all propositional atoms. A formal description is given in the next chapter.

Reinfrank et al. [14] have tried to formalize the concept of justified belief(beliefs or data that have a proof in the TMS, starting from assumptions and using the justifications) in terms of *Autoepistemic logic*(AEL).

2.4.2 Autoepistemic Logic

Autoepistemic Logic was developed by Moore[12], as a means of formalizing commonsense reasoning as a model of an ideally rational agent's reasoning about his own beliefs. Konolige[8] suggests that defaults can be represented in terms of reasoning about self-belief. For example, the default statement

Birds normally fly.

can be expressed, in terms of self-belief, as

If I know that x is a bat, then I'll assume x flies **unless** I have information to the contrary.

So in the presence of conflicting information, the rule is inapplicable, because of the "unless" clause, thus AEL is able to capture nonmonotonicity.

In AEL, the self-belief is represented by using a modal operator L. The sentence Lp is means: "the sentence p is one of my beliefs". Then the AEL version of the sentence above is the schema

$$LBx \bigwedge \neg L \neg Fx \supset x$$

.

The semantics of AEL is given in terms of sets of sentences T that are derivable from the premises and are *stable* with respect to self-belief, that is, a sentence p is in T iff Lp is in T. For example, let us assume that the initial set of premises are $\{\neg Lp \rightarrow q, \neg Lq \rightarrow p\}$. The belief sets derivable from the premises are either p or q(not including the premises and the sentences of the form Lp, LLp,...). If p is in the belief set, Lp is also;so q can not be derived from the premises and is not in the belief set. Similarly for q.

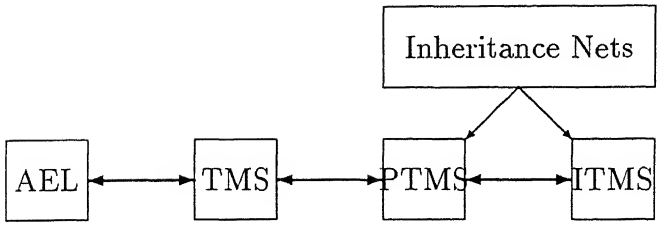


Figure 2.19: Architecture of the solution.

2.5 Design of the Solution

The problem of computing skeptical inheritance in the nets and showing it to be a subproblem of a larger framework of commonsense reasoning is solved by translating the nets into modified TMS's.

The correspondence between AEL and TMS established in [14] means that if we translate the notion of inferencing in nets into a TMS formalism, we will be assured of the fact that this can also be translated into AEL. This way the semantics of the derivation of inheritance information in the nets will be provided by the semantics of AEL. This will also show how the problem of inheritance can be embedded in the general framework of commonsense reasoning.

We draw insight from [1] where Brewka has given a TMS-based theory of inheritance . The work follows the architecture given in figure 2.19.

Since working in the TMS is cumbersome and the justifications generated are large, Brewka [1] proposes a modified TMS(ITMS or Inheritance TMS) which exploits the regularity of information in the nets and is much simpler. The problem with [1] is that computes only credulous extensions. There is no constructive procedure for computing these extensions. The intricacies introduced by cyclic and mixed links have not been dealt with.

We follow the same basic architecture but now the TMS is replaced by a PTMS standing for "Parametrized TMS", where a set of justifications are added to a core set depending upon a parameter. These set of justifications express whether to prefer a node A or $\neg A$ in the presence of an ambiguity about A. The equivalent translation to ITMS also includes this parameter which characterizes one of the extensions. Then we devise a way to associate a set of these extensions in which an inference holds. This gives us a method for determining skeptical inheritance. From the connections 1 and 2 in figure 2.19 we are assured of getting a semantics for the system proposed in terms of AEL. For cyclic

nets, we generalize the definition of specificity(which collapses to the *more specific* case when nets are acyclic). To introduce strict links we generalize the usual definition of *children*[1] of a node in the net.

Thus we are able to handle the most general inheritance nets by these incremental generalizations on acyclic nets.

[1]Let the inheritance net be $\mathcal{G}$. The positive children of a node n are $pos(n) = \{x \mid x \rightarrow n \in \mathcal{G}\}$. The negative children of a node n are $neg(n) = \{x \mid x \not\rightarrow n \in \mathcal{G}\}$.

3 A TMS-based Theory of Inheritance

In this chapter, we present a simple theory of inheritance for an acyclic defeasible inheritance nets. In section 3.1 , we give the formal description of a TMS. In section 3.2, we describes a modified TMS, called the PTMS and give rules to translate the nets into the PTMS. In section 3.4, we describe a simpler PTMS, called the ITMS and give the equivalence between ITMS and PTMS. Issues in computation of skeptical inheritance and an algorithm for this purpose are given in the next two sections.

3.1 Formal Description of a TMS

This description is from [1].

A TMS network Γ = (N, Σ) consists of a set of nodes N and a set of justifications Σ. Each justification is of the form

$$< A_1, \ldots, A_m \mid B_1, \ldots, B_n \rightarrow C > .$$

Intuitively, if each A_i is IN and each B_j is OUT, then C is IN. The TMS computes an IN/OUT labeling for the nodes in N. The justifications act as constraints on the acceptable labelings.

For all the definitions below, let Γ = (N, Σ) be the TMS network.

Let L: $N \rightarrow \{IN, OUT\}$ be a labeling.

Definition 11 A justification $< A_1, \ldots, A_n \mid B_1, \ldots, B_m \rightarrow C >$ is *valid* for C in L iff
$L(A_i) = IN, 1 \leq i \leq n$, and
$L(B_j) = OUT, 1 \leq j \leq m$.

Definition 12 The labeling L is *closed* iff
if $\forall\sigma \in \Sigma \ni \sigma$ is valid in L for a node C, then $L(C) = IN$.

Definition 13 The labeling L is *grounded* iff there is a linear ordering $(N_1, N_2, \ldots)$ of the set {A| A∈ N, L(A) = IN} such that every node N_j has a valid justification
$< N_{j_1}, \ldots, N_{j_u} \mid M_1, \ldots, M_v \longrightarrow N_j >$ such that $j_i \leq j (1 \leq i \leq n)$.

Groundedness basically implies that any IN node has a non-circular proof through a sequence of valid justifications.

The labeling representing justified belief sets are those that are *closed* and *grounded.* It follows from the definitions then that for such a labeling, a node is labeled IN (any acceptable belief) iff it has a proof(i. e. , sequence of valid justifications leading upto the node). The correspondence between TMS and AEL [14] is established through the justified belief sets. Hence, whenever we refer to TMS-labelings, we always mean "closed and grounded labelings".

3.2 A Parametrized TMS(PTMS)

A PTMS is a modification of the conventional TMS which has been described in the previous section.

Definition 14 A PTMS is a triple (N, Σ, Π), where
N is a set of nodes,
Σ, a set of justifications, and
Π, a subset(possibly empty) of N. Π is called a *preference set.*

Let $\alpha(\Pi)$ be a set of justifications built in some way, based on the information in Π. $\alpha(\Pi)$ is called the *induced set on* Π

Definition 15 A labeling L: $N \rightarrow \{IN, OUT\}$ is closed and grounded for a PTMS = (N, Σ, Π) iff the labeling L is closed and grounded for the TMS = $(N, \Sigma \cup \alpha(\Pi))$.

Thus, with different preference sets Π, we get different labelings for the same set of nodes and justifications. We denote, by L^{Π}, a labeling when the preference set is Π. The role of preference sets will be apparent when we the translate inheritance nets to a PTMS.

3.2.1 Description of the PTMS

We have the following nodes for the PTMS.

1. *Ordinary Nodes:* For each node A in the net, we have a corresponding node A in the PTMS.

2. *Negative Nodes:* Since the TMS can not handle negation, we have negative nodes of the type $\neg A$. To minimize the number of unnecessary negative nodes, we introduce negative nodes only when there is a negative link to A in $\mathcal{G}$.

   An ordinary/negative node, say A, actually stands for some path $\pi(S.\sigma.[\neg]A)$ in the net. This intuitive correspondence is proved later.

3. *Abnormal(Ab) nodes:* Nodes of type $abCA[\neg]B$ express the fact that the path $\pi(S.\tau.A{\rightarrow}B)$ (respectively, $\pi(S.\tau.A{\not\rightarrow}B)$) has been preempted by another path $\pi(S.\delta.C{\not\rightarrow}B)$ (respectively, $\pi(S.\delta.C{\rightarrow}B)$) . The node S is the focus node.

4. *Specificity Nodes:* The nodes of type $spA[\neg]$B express the fact that relative to the focus node, A is more specific than $[\neg]$B. A specificity node spAB(resp. spA$\neg$B) is introduced in the PTMS, only when there is a positive(respectively, negative) path $\sigma(S.\tau.A\sigma.B)$(respectively, $\sigma(S.\tau.A\sigma.\neg B)$)in $\mathcal{G}$.

5. Conflict Nodes: The nodes of type cntA denote that there is a conflict at A which has not been resolved. This is where the preference set comes into the picture. If cntA is IN(asserting the conflict at A), and A(respectively, $[\neg]$A) is in the preference set, then we allow A(respectively, $[\neg]$A) to be IN. This then captures the idea of resolving the conflict by "choice".

We call the ordinary and negative nodes the *net nodes* and other nodes the *external nodes*, since it is only the former kind of nodes about which we are trying to infer the inheritance relations. The external nodes are just for capturing the "hidden" inheritance information in the hierarchy.

For the focus node A, we have a Justification of the form $< \mid \longrightarrow A >$, which is called a *premise justification.* Since the IN-list and the OUT-list of this justification is empty, it always makes the focus node IN. All other justifications are built through the translation.

The preference set Π is concerned only about the internal nodes, since these are the nodes at which conflicts are to be resolved in the inheritance hierarchy. Π is such that for no internal node $C \in N$, both C and $\neg C$ are in N. This means that in case of a conflict over a node C, one chooses either C or $\neg C$.

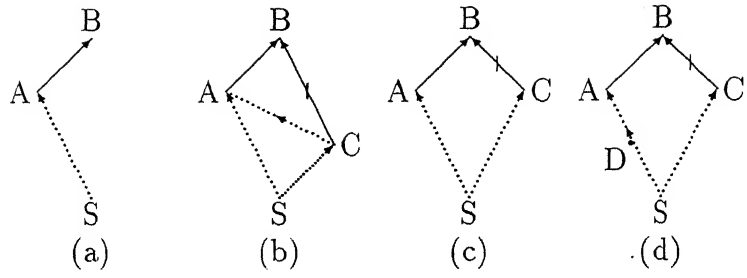


Figure 3.1: Situations during inference in nets.

3.2.2 Translation

The justifications reflect the bottom-up construction of proofs i. e. if we have a proof of, say, A , then the next proof is done for a node B, such that there is an edge $A \rightarrow B$(or, $A \not\rightarrow B$) in $\mathcal{G}$ (figure 3.1. (a)) .

Let $\mathcal{G}$ be the inheritance net with the nodes V. Compute the following PTMS node sets. $M = \{x \mid x \in V\}$,

$\overline{M} = \{\neg B \mid A \not\rightarrow B \in \mathcal{G}\}$,

$Ab = \{abDAB \mid A \rightarrow B$ and $D \not\rightarrow B \in \mathcal{G}\} \cup \{abDA\neg B \mid A \not\rightarrow B$ and $D \rightarrow B \in \mathcal{G}\}$,

$Sp = \{spAC \mid$ there is a positive path $\pi(S.\sigma.A.\tau.C) \in \mathcal{G}$, τ may be empty$\}$,

$CNT = \{cntA \mid A \in M\}$,

Let $N = M \cup \overline{M} \cup Ab \cup Sp \cup CNT$.

Σ is the smallest set of justifications such that

Rule 1: If the focus node is N, then the premise justification $< \mid \longrightarrow N > \in \mathcal{G}$.

Since we have in mind a closed and grounded labeling, if a node A is IN, it will mean we have a proof for A starting from the assumptions. The only assumption we make is the premise justification which makes the focus node IN always since the In-list and the OUT-list are empty for the premise justification. So, proofs are always relative to the focus node. Note that there is only one such premise justification, denoting that there is only one focus.

For the inductive steps of the proofs, we have the following observations. The idea is to capture these notions through the PTMS justifications. Figure 3.1 gives graphic details of the situations that entail the justifications.

Consider the case when we have a valid path $S.\sigma_1.A$ and there is an edge $A \rightarrow B$ in $\mathcal{G}$. The other case when the edge $A \not\rightarrow B$ is in $\mathcal{G}$ is symmetrical.

If B has no negative children, i. e. there is no node C such that $C \not\rightarrow B$ is in $\mathcal{G}$, then there are no negative paths from the focus to B . So the path $\pi(S.\sigma_1.A \rightarrow B)$ should hold. (See figure 3.1.(a)). Hence we have the following rule:

Figure 3.2: explicit contradiction in inheritance nets.

Rule 2.1: if $A \rightarrow B \in \mathcal{G}$ and B has no negative children, then $< A \mid \longrightarrow B > \in \Sigma$.

If B has negative children, then the problem of resolution among contradictory paths comes into play. If A is also a negative child, i. e. both $A \rightarrow B$ and $A \not\rightarrow B$ are in $\mathcal{G}$, there is an explicit contradiction in the net, which is given *a priori*. See figure 3.2. The paths $\pi(S.\tau.A \rightarrow B)$ and $\pi(S.\tau.A \not\rightarrow B)$ do not preempt each other since no one represents a more specific argument than the other. Also, we should not report this explicit contradiction through a contradictory node $cntB$, since the contradictory nodes actually report a conflict that can be resolved through choice, while the explicit contradictions cannot be resolved in any way. We do the following:

- In case of an explicit contradiction at B (when none of the contradictory paths are preempted by some other path in $\mathcal{G}$, of course) we report both B and $\neg B$. This can be done by imposing that A is not in pos(B) when $A \not\rightarrow B$ is considered, and A is not in neg(B) when $A \rightarrow B$ is considered. Then the explicitly contradictory paths cannot preempt each other. Preemption of other paths, however, remains unaffected.

- We can have a special node called *INCONSISTENT(P)* in the PTMS. When there is an explicit contradiction at the node P, this node is IN. In that case, one can go about modifying the net. This can be done by adding to the PTMS the justifications of form

  $$< P, \neg P \longrightarrow INCONSISTENT(P) >$$

  for all nodes P such that there are edges $Q \rightarrow P$ and $Q \not\rightarrow P$ are in $\mathcal{G}$. But since we are not interested in the recovery procedure for such explicit contradictions, we do not add these justifications to the present PTMS.

Let C be a negative child (C is never A), such that there is a valid path $\pi(S.\sigma_2.C)$. (The negative nodes that have no valid paths are useless for the inference at B.)

Two cases may arise.

Case 1 : The path $\pi(S.\sigma_2.C \not\rightarrow B)$ preempts the path $\pi(S.\sigma_1.A \rightarrow B)$ i. e. $C <_S A$. (Figure 3.1.(b)). This is captured by the following rule:

Rule 2.2: if $A \rightarrow B \in \mathcal{G}$ and C is a negative child of B , then
$< spCA \mid \longrightarrow abCAB > \in \Sigma$ if $spCA \in N$.

Case 2: The path $\pi(S\sigma_2.C \not\rightarrow B)$ does not preempt the path $\pi(S.\sigma_1.A \rightarrow B)$ but is itself not preempted. This means there is a conflict at B which is unresolved. (Figure 3.1.(c)) Hence the rule:

Rule 2.3: if $A \rightarrow B \in \mathcal{G}$ and C is a negative child of B , then
$< A, C \mid abCAB, abAC\neg B \longrightarrow cntB > \in \Sigma$.

Validity of this justification means the precondition in step 2.(c) of the definition of permission for $\pi(S.\sigma_1.A \rightarrow B)$ is true. For the validity of the path then we must have B in the constraint (preference) set for $\mathcal{G}$.

If the path $S.\sigma_1.A \rightarrow B$ is not preempted and there is no conflict, then it is a valid path.

Rule 2.4: if $A \rightarrow B \in \mathcal{G}$ and C is a negative child of B , then
$< A \mid abCAB, cntB \longrightarrow B > \in \Sigma$.

Validity of this justification implies that the step 2.(b) of the definition of permission for $\pi(S.\sigma_1.A \rightarrow B)$ is true, hence the path is valid.

Symmetrically, considering the edge $A \not\rightarrow B$, we have the following rules:

Rule 3.1: if $A \not\rightarrow B \in \mathcal{G}$ and B has no positive children, then
$< A \mid \longrightarrow \neg B > \in \Sigma$.

Rule 3.2: if $A \not\rightarrow B \in \mathcal{G}$ and C is a positive child of B , then
$< spCA \mid abCA\neg B \longrightarrow \in > \Sigma$ if $spCA \in N$.

Rule 3.3: if $A \not\rightarrow B \in \mathcal{G}$ and C is a positive child of B , then
$< A, C \mid abCA\neg B, abACB \longrightarrow cntB > \in \Sigma$.

Rule 3.4: if $A \not\rightarrow B \in \mathcal{G}$ and C is a positive child of B , then
$< A \mid abCA\neg B, cntB \longrightarrow \neg B > \in \Sigma$.

In case of a conflict at a node B, a choice is made between B or $\neg B$ as is in the preference set Π. This is done through the induced set on Π. The path $\pi(S.\sigma_1.A \rightarrow B)$ is valid if when there is a conflict then at B , then B is chosen. Hence, we have the following rule.

Rule 4: The induced set on Π, $\alpha(\Pi)$ is given as:
if $K \in \Pi$ then $< cntK \mid \longrightarrow K > \in \Sigma$
if $\neg K \in \Pi$ then $< cntK \mid \longrightarrow \neg K > \in \Sigma$

Now, see figure 3.1.(a) again. If the path $S.\sigma_1.A \rightarrow B$ is valid, then $A <_S B$. Figure 3.1.(d) shows when specificity relation holds between two non-adjacent nodes. It expresses the fact that when $\pi(S.\tau_1.D.\tau_2.A \rightarrow B)$ is valid, $D <_S B$. The following rules

capture the above ideas.

Rule 5: if $A \rightarrow B \in \mathcal{G}$ then

5.1: if B has no negative children, then $< A \mid \longrightarrow spAB > \in \Sigma$ and $\forall D \ni spDA \in N, < spCA \mid \longrightarrow spCB > \in \Sigma$.

If B has negative children, then for every negative child C (figure 3.1.(d)),

5.2. $< A \mid abCAB, cntB \longrightarrow spAB > \in \Sigma$.

5.3. $< A, cntB, B \mid \longrightarrow spAB > \in \Sigma$.

5.4. $\forall D \ni spDA \in N, < spDA \mid abCAB, cntB \longrightarrow spDB > \in \Sigma$.

5.5. $\forall D \ni spDA \in N, < spDA, cntB, B \mid \longrightarrow spDB > \in \Sigma$.

Validity of the justifications from rules 5.2 and 5. 4 correspond to the validity of a path $\pi(S.\tau_1.F \rightarrow B)$, which has the node C on it, by step 2.(b) of the permission relation. F stands for the penultimate node of this path. It may be C itself.

Validity of the justifications from rules 5.3 and 5.5 means, for the path $\pi(S.\tau_1.F \rightarrow B)$, which has C on it, step 2.(c) of the permission relation is true . This is because, (1) since $cntB$ is true, the precondition holds and (2) B must have come from the induced justification $< cntB \mid \longrightarrow B >$, so B is in the preference set.

In either case, the validity of the justifications correspond to the fact that C is more specific than B.

Validity of justifications from rules 5.2 and 5.3 imply that the path $\pi(S.\sigma_1.A \rightarrow B)$ is valid by steps 2.(b) and 2.(c) respectively, of the definition of permission for the path. Then, validity of these justifications correspond to the fact that $A <_S B$.

The correspondence between validity of justifications and validity of paths in the net presume

1. a node A is IN is equivalent to the fact that a path $\pi(S.\sigma.A)$ is valid in $\mathcal{G}$.
2. $spAB$ is IN is equivalent to the fact that C is more specific than B in $\mathcal{G}$.
3. the above assumptions are true at least for the antecedents of the justifications.

The formal proof of the correspondence is an easy result of the discussion above. Before stating the result, we mention a shortcoming of the translation to PTMS and illustrate this with an example.

The above translation generates a large number of justifications. This is due to the fact that

- explicit information about specificity has to to be represented by explicit nodes (of type spAB) and explicit justifications, and

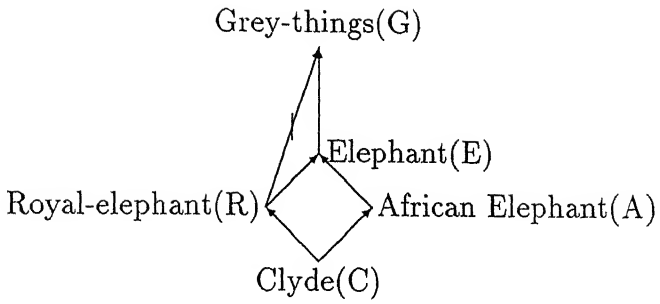


Figure 3.3: The modified royal elephant example.

- in case of unresolved conflicts we will have to have justifications of the induced set to resolve the conflict.

In the next section, we describe the modified TMS(the ITMS or Inheritance TMS), which exploits the regularity in the specificity and conflict information and represents the justifications in a compact form, thus reducing the number of justifications greatly.

Example 3.1 Consider the net in figure 3.3. We follow the abbreviations for the nodes noted in the parentheses.

With an empty preference set we have the PTMS (N, Σ, Π), where

N = {C, R, A, E, G, ¬ G, abREG, abER¬ G, spCR, spCA, spRE, spAE, spEG, spCE, spCG, spRG, spAG, cntG},

$\Sigma =$

{

$< C \mid \longrightarrow R >$,

$< C \mid \longrightarrow A >$,

$< R \mid \longrightarrow E >$,

$< A \mid \longrightarrow E >$,

$< E \mid abREG, cntG \longrightarrow G >$,

$< spRE \mid \longrightarrow abEG >$,

$< E, R \mid abREG, abR\neg G \longrightarrow cntG >$,

$< R \mid abR\neg G, cntG \longrightarrow \neg G >$,

$< R, E \mid abREG, abR\neg G \longrightarrow cntG >$,

$< C \mid \longrightarrow spCR >$

$< C \mid \longrightarrow spCA >$

$< R \mid \longrightarrow spRE >$

$< A \mid \longrightarrow spAE >$

$< E \mid abREG, cntG \longrightarrow spEG >$

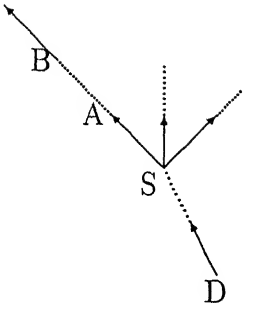


Figure 3.4: Only those nodes that are "above" the focus S are considered for computing inheritance.

$< E, cntG, G \mid \longrightarrow spEG >$
$< E, spAE \mid abREG, cntG \longrightarrow spAG >$
$< E, spRE \mid abREG, cntG \longrightarrow spRG >$
$< E, spCE \mid abREG, cntG \longrightarrow spCG >$
$< E, spAE, cntG, G \mid \longrightarrow spAG >$
$< E, spRE, cntG, G \mid \longrightarrow spRG >$
$< E, spCE, cntG, G \mid \longrightarrow spCG >$
}
$\Pi = \{\}$.

Since the preference set is empty, the induced set on it is also empty.

Let us find out the labels of the nodes.

C is IN, from the premise justification. Hence, the nodes A, R, E, spCR, spCA, spRE, spAE, spCE and abREG are IN. The node abR¬G is OUT, since there is no valid justification for it. From this we get G is OUT and ¬G is IN. To summarise, with Clyde as the focus, we find it is a Royal elephant, African elephant, Elephant and Non-grey.

The intuitive meanings attached to the justifications introduced by the translation is made concrete by the following lemma.

The proof of the lemma is by induction on the topological order of the nodes of $\mathcal{G}$. If a node A is topologically earlier than another node B , then we represent it as $A \leq_T B$. Note that, since the net is acyclic, the topological order is well-defined. Also, since the focus can inherit only from those nodes that are strictly later in the topological order (i. e. above the focus in the net), we can safely ignore the nodes D such that $D \leq_T S$(See figure 3.4).

Lemma 1 Let $\Gamma = (N, \Sigma, \Pi)$ be the translation of an acyclic defeasible net $\mathcal{G}$ with nodes M . If L_Π is a closed and grounded labeling for Γ, then the following is true:

1. $L_\Pi(spCA) = IN \Leftrightarrow C <_S A$ in $\mathcal{G}$.
2. $L_\Pi(A) = IN \Leftrightarrow$ some path $\pi(s.\tau.A)$ is permitted by $\mathcal{G}$.

Proof :

Induction base: Consider any node B such that $S \rightarrow B \in \mathcal{G}$. $\mathcal{G} \triangleright S \rightarrow B$, since this is an explicit relation of the focus . From definition of specificity, it follows that $S <_S B$. Correspondingly, in the PTMS, we have justifications $< S \mid \longrightarrow B >$(from rule 2.1) and $< S \mid \longrightarrow spSB >$(from rule 5.1). Hence, B is IN, and $spSB$ is IN. So the lemma is true for the nodes B which are immediately next to the focus in topological order.

Consider any other node B.

Induction hypothesis: For all $A <_T B$, the lemma is true, i. e.

1. $L_\Pi(spCA) = IN \Leftrightarrow C <_S A$ in $\mathcal{G}$, if for some node D, $spCA \in N$, i. e. some path $\pi(S.\tau_1.C.\tau_2.A)$ is valid in $\mathcal{G}$,where τ_2 may be empty in which case $C \rightarrow A \in \mathcal{G}$.
2. $L_\Pi(A) = IN \Leftrightarrow$ some path $\pi(s.\tau.A)$ is permitted by $\mathcal{G}$.

Induction step for proof of (1)

($\Rightarrow$): Let $L_\Pi(spCB) = IN$ for some node C . Since the labeling is grounded, some justification in Σ is valid for spCB. These justifications may be due to rules 5. 1, 5. 2, 5. 3, 5. 4 and 5. 5 and may be one of the following:

1. $< C \mid \longrightarrow spCB >$, if C has no negative children, and $C \rightarrow B \in \mathcal{G}$,
2. $< spCA \mid \longrightarrow spCB >$, if C has no negative children,and $A \rightarrow B \in \mathcal{G}$,
3. $< C \mid abDCB, cntB \longrightarrow spCB >$, where $C \rightarrow B \in \mathcal{G}$ and D is a negative child of B,
4. $< C, cntB, B \mid \longrightarrow spCB >$, where $C \rightarrow B \in \mathcal{G}$ and there is a negative child of B.
5. $< spCA \mid abDAB, cntB \longrightarrow spCB >$, where $A \rightarrow B \in \mathcal{G}$ and D is a negative child of B, and
6. $< spCA, cntB, B \mid \longrightarrow spCB >$, where $A \rightarrow B \in \mathcal{G}$ and there is a negative child of B.

If $spCB$ is IN through justifications 1 or 2, according to the conditions, B has no negative children, hence $\sigma.F \rightarrow B$ is valid, since then there is no question of preemption or conflict. So, by definition of specificity $C <_S B$ (because it is on a valid path that enables B).

Then, we have

- If $spCB$ is IN through justifications 3 or 4, then the path $\pi(S.\sigma.C \rightarrow B)$ is valid (refer rule 5. 2 and 5. 3), and
- If $spCB$ is IN through justifications 4 or 5, then the path $\pi(S.\tau_1.C.\tau_2.A \rightarrow B)$ is valid (refer rules 5. 4 and 5. 5).

  Hence, in either case, $C <_S B$.

($\Leftarrow$):Let $C <_S B$. Then there is a permitted path from the focus to B which has C on it.

Let the path be $\delta = \pi(S.\sigma.C \rightarrow B)$. The subpath $\pi(S.\sigma.C)$ is permitted.So, by induction hypothesis, C is IN.

Regarding the negative children of B, we have two cases.

- B has no negative children. Then, we have justification 1 in Σ (from rule 5. 1). Hence $spCB$ ia IN.
- B has negative children. Consider a particular negative child , say D. Then we have the justifications 3 and 4. Since δ is valid, either step 2.(b) or step 2.(c) of the permission relation is true, hence one of the justifications is valid. So, $spCB$ is IN.

Let the path be $\delta = \pi(S.\tau_1.C.\tau_1.A \rightarrow B)$. By induction hypothesis, A is IN and $spCA$ is IN. It can be easily shown that $spCB$ is IN (one has to consider the justifications 2, 5 and 6 now).

Inductive step for proof of (2).

The inductive proof technique exactly follows the one described in the proof of (1). Hence we omit the details for the proof of this part.

□

The correspondence between the abnormal nodes in PTMS and the notion of preemption in the net is formally established now.

Corollary 1 $L_{\Pi}(abCAB) = IN \Leftrightarrow$ each path $\pi(s.\tau, A \rightarrow B)$ is preempted by a path $\pi(S.\tau_1.C \not\rightarrow B) in \mathcal{G}$.

Proof : $Let L_{\Pi}(abCAB) = IN$. Then there is a negative child C of B. From groundedness, $spCA$ is IN . From the lemma 1, $C <_S A$. Hence by definition of preemption, any path $\pi(s.\tau, A \rightarrow B)$ is preempted by $\pi(s.\tau_1, C \not\rightarrow B)$.

The proof of the *if* part is similar and direct.

□

Given the choices at all the conflicts, we can say for sure whether a node holds or not in the net. If the choices are for a subset of conflicts, the ambiguous nodes without choices are OUT (do not hold). Hence, we always get a single labeling for the nodes of PTMS.

Lemma 2 There is a unique closed and grounded labeling for the PTMS $\Gamma = (N, \Sigma, \Pi)$.

Proof : From the correspondence between the paths in the net and the labeling of nodes established through the above lemmas, we can define what we call a *measure* of the nodes of the PTMS.

measure(A)= the length of the longest path from the focus to A.

measure(spCA)= the length of the longest positive path from the focus to A through the node C.

The proof of uniqueness of labels is by induction on the measure of the nodes. When measure(N)= 0, then N is the focus node, whose label is IN always, hence unique.

From the structure of the justifications, by groundedness, we see that the label of any node B follows from the labels of its children and from nodes of type spCA, where C and A are the children of the node uder consideration. Since for any child D of B, measure(D) ¡ measure(B) and also measure(spCD) ¡ measure(B). By induction hypothesis, we have that the labels of the nodes D and nodes spCD have unique labels. Hence, by closedness, the label of B is also unique.

□

By the corollary above, whenever we mention a labeling for a PTMS, we will refer to this unique labeling.

3.3 Preference Sets

The introduction of the preference set in the PTMS leads to certain interesting behaviour of the TMS when the preference set is used differently in presence of conflicts.

- If there are conflicts and the preference is taken to be empty, the induced set is empty. So at the ambiguous node A, PTMS labels cntA IN. This means A is OUT, so that all the subsequent justifications depending upon A become invalid. The system then behaves like an "ambiguity blocking" formalism. See figure 2.7. Here, cntE is IN and E is OUT. Since spED is not among the nodes, abDF is OUT. Similarly, spDE is also OUT. Therefore, both abDF and abE$\neg$F are OUT. Subsequently, we have F IN.

- If the preference set Π always chooses in favour of a node A(and not in favour of $\neg A$) in the presence of a conflict, the system behaves like "ambiguity propagating", because then for every ambiguous node A, we have cntA IN, and hence from the induced set on Π, we have A IN. So, we are able to mark the ambiguous nodes and also are able to assume that they hold, thus allowing the nodes to participate in further derivations.

3.3.1 Nature of Preference Sets

A preference set is just a set of nodes such that no node with both positive and negative polarity may be in the set. Given any set of nodes P, one can construct many preference sets by choosing any subset P with either positive or negative choice at each node.

If a node A holds in a preference set P that resolves all the conflicts , it automatically holds in the preference sets that are supersets of P, because then the extra nodes are superfluous. We say that these supersets are *redundant* with respect to P for the node under consideration. We also call a choice at a node *redundant* with respect to a preference set if no choice at that node is in the preference set.

In general, a preference set may have a choice at each of the nodes of N. We call such a preference set *naive*, since it is always redundant(because the focus node and the nodes immediately next to the focus hold always and there is no ambiguity about them). We can define a naive preference set as:

$$\{\text{either } P \text{ or } \neg P \mid P \in N\}.$$

There are exactly $2^{|N|}$ such naive preference sets. The purpose of introducing naive preference sets is to facilitate the computation of skeptical inheritance.

In an extension, we have a choice (either C or $\neg C$) at every ambiguous node C. A preference set is said to be *cautious* if it has a choice at each ambiguity. Formally,

Definition 16 If for any internal node A, the PTMS $\Gamma = (N, \Sigma, \Pi)$ labels cntA IN, then the preference set Π is said to be *cautious* if either A or $\neg A$ is in Π.

A cautious preference set P stands for exactly one extension X. Since the cautious set resolves all the conflicts, any superset Q of P will be redundant with respect to to P, hence Q also stands for the same extension X. We can extend the cautious set P to a set of naive sets by adding either C or $\neg C$ for each node C in $N \setminus P$. Each such naive set stands for the same extension X.

3.4 Inheritance TMS

The PTMS employs many supplementary nodes (specificity, conflict and Abnormal nodes) along with the nodes of interest, i. e. , the ordinary and the complementary nodes. The interconnection between justifications is complex. In order to express intuition in a clearer and simpler manner, we introduce the ITMS which has complex labels and has a different meaning for the justifications.

3.4.1 Description of the ITMS

An ITMS is a triple (N, Σ, Π), where N is a set of nodes, Σ is a set of justifications, and Π is a possibly empty subset of N, called the preference set.

In an ITMS, the label IN of PTMS is replaced by a set of nodes. The meaning of the new form of labeling is as follows.

If a node P is labeled X, where X $\subset$ N, then

1. P is in the current set of beliefs(P is IN), and
2. $\forall P_i \in X, P_i$ is more specific than P. One can see that this labeling replaces the nodes of form spP_iP.

The label of a node A may also be empty. Then, it will mean that A is IN and there is no node more specific than A . One may note that this is true only for the focus node.

A labeling for an ITMS means assigning labels to the nodes of the ITMS. With this labeling we define specificity in ITMS as below.

Definition 17 [Specificity]

A node A is more specific than a node B in the ITMS, written $A <_L B$, under the label L iff $A \in L(B)$.

The set of nodes in the ITMS consists of only the internal nodes, i.e. the ordinary nodes and the complementary nodes. The auxiliary nodes of PTMS are done away with. The nodes represent, as in PTMS, proofs in the net with respect to the focus.

The set of justifications Γ contains only two type of justifications: one premise justification of type $< : \{\} \longrightarrow [\neg]A >$ and justifications of type

$$< A : \{X\} \longrightarrow [\neg]B >$$

where A and [¬]B are nodes and X is a set of nodes denoting the positive children of B if A is a negative child of B and negative children of B if A is a positive child of B. A is called the antecedent of the justification and B its consequent. When the label of A contains a node C which is in X , it means that there is a child of opposite polarity to A which is more specific than A . Hence, the proof of $[\neg]B$ through A is preempted.

In the ITMS, the notion of preemption which is relative to the paths is captured through a new relation *overrides* which is defined below.

Definition 18 [Overriding]

Let (N, Σ, Π) be an ITMS and let L^{Π} be a labeling for this ITMS.

Let $X = \text{pos}(B)$ and $Y = \text{neg}(B)$. A node A in pos(respectively, neg) is overridden for B(with respect to the focus) by a node $C \in Y$(respectively, X) iff $C <_L A$.

With the definition of overriding above we define the notion of validity, closedness and groundedness in the ITMS $\Gamma = (N, \Sigma, \Pi)$. The labelings are denoted by L^{Π}. Also, at times, when $L^{\Pi}(A) \neq \text{OUT}$, we write it as $L^{\Pi}(A) = \text{IN}$.

Definition 19 [Validity]

An ITMS justification $\sigma \in \Sigma$ is *valid* for a node [¬]B iff

either

1. B is the focus node i. e. $\sigma = < : \{\} \longrightarrow [\neg]B >$, or

2. A child of B, say A, is IN, no other child of B overrides A for B and there is no conflict i. e. $\sigma = < A : \{X\} \longrightarrow [\neg]B >$, $L^{\Pi}(A) = IN$ and $\forall P \in Y \ni L^{\Pi}(P) = IN$, A is not overridden for B by P and A overrides P for B, or

3. there is a conflict at B and [¬]B is chosen i. e. $\sigma = < A : \{X\} \longrightarrow [\neg]B >$, $L^{\Pi}(A) = IN$ and $\exists P \in Y \ni L^{\Pi}(P) = IN$, A is not overridden for B by P and P is not overridden for B by P and [¬]"B" $\in \Pi$.

Validity of a justification $< A : \{X\} \longrightarrow B >$ means from a valid proof of A we can build a valid proof of B .

Definition 20 A labeling $L^{\Pi} : N \Longrightarrow \mathcal{P}(N) \cup \{OUT\}$ is *closed* iff

1. if $< : \{\} \longrightarrow B > \in \Sigma$ is valid in L^{Π} then $L^{\Pi}(B) \neq$ OUT, and

2. if $< A : \{X\} \longrightarrow B > \in \Sigma$ is valid in L^{Π} then $L^{\Pi}(A) \cup \{A\} \subseteq L^{\Pi}(B)$.

Closedness in ITMS means two things. First, if there is a valid justification for some node B , then the node is IN(i. e. there is a valid proof for B). Second, if there is a valid proof of B through a valid proof of A , then A is more specific than B and the nodes are more specific than A are also more specific than B .

Definition 21 A labeling $L^{\Pi} : N \Longrightarrow \mathcal{P}(N) \cup \{OUT\}$ is *grounded* iff there is a linear ordering
$((A_1, Y_1), \ldots)$ of the set $\{(AY) \mid A \in N,\ L^{\Pi}(A) = Y \neq OUT\}$ such that for every pair (A_i, Y_i):
if $Y_i = \{\}$, then $< : \{\} \longrightarrow A_i > \in \Sigma$, else
$\forall B \in Y_i, \exists \sigma = < A_h : \{X\} \longrightarrow A_i > \in \Sigma \ni h < i, \sigma$ is valid in L^{Π} and $B \in Y_h \cup \{A_h\}$.

Groundedness in ITMS means two things. First, if some node B is IN, then it has a non-circular proof through a sequence of valid justifications. Second, the antecedents of these valid justifications make up all the nodes in the label of B. As before, we consider only closed and grounded labelings for any ITMS.

The translation of an inheritance net to the ITMS is simple and direct.

Definition 22 [Translation]
Let $\mathcal{G}$ be an inheritance network with nodes M.
The positive children of a node $N \in \mathcal{G}$ are $pos(N) = \{B \mid B \rightarrow N \in \mathcal{G}\ \}$

The negative children of a node $N \in \mathcal{G}$ are $neg(N) = \{B \mid B \not\rightarrow N \in \mathcal{G}\}$

Then the ITMS translation of $\mathcal{G}$ with the preference set Π is the ITMS network (N, Σ, Π), where

$N = M \cup \{\neg B \mid A \not\rightarrow \mathcal{G}\}$, and

Σ is the smallest set of justifications such that

1. if $A \rightarrow B \in \mathcal{G}$ then $< A : \{X\} \longrightarrow B > \in \Sigma$, where $X = neg(B) \setminus \{A\}$
2. if $A \not\rightarrow B \in \mathcal{G}$ then $< A : \{X\} \longrightarrow \neg B > \in \Sigma$, where $X = pos(B) \setminus \{A\}$

Example 3.2 The ITMS translation of the net in fig 3.3 is given below.

1. $< C : \{\} \longrightarrow R >$,
2. $< C : \{\} \longrightarrow A >$,
3. $< R : \{\} \longrightarrow E >$,
4. $< A : \{\} \longrightarrow E >$,
5. $< R : \{E\} \longrightarrow \neg G >$,
6. $< E : \{R\} \longrightarrow G >$,

It is easy to see that the justifications 1, 2 are valid for R and A. Hence $C \in L(R)$ and $C \in L(A)$. Since R and A are IN, the justifications 3 and 4 also are valid, so $C, R, A \in L(E)$. Since $R \in L(E)$, R overrides E for G . So, justification 5 is valid and justification 6 is not valid. Note that there is no conflict because E does not override R for G. Hence $\neg G$ is IN and G is OUT.

The connection between inference in the inheritance graphs and the labelings in their ITMS translations is established through the following lemma. It is very similar to the corresponding lemma in case of PTMS. Since the definition of validity of justifications and that of paths in the net matches, we have omitted details in the proof.

Lemma 3 Let $\Gamma = (N, \Sigma, \Pi)$ be the translation of an acyclic defeasible net $\mathcal{G}$ with nodes M . If L_Π is a closed and grounded labeling for Γ, then the following is true:

1. $C <_I A \Leftrightarrow C <_S A$ in $\mathcal{G}$.
2. $L_\Pi(A) \neq OUT \Leftrightarrow$ some path $\pi(s.\tau.A)$ is permitted by $\mathcal{G}$.

Proof :

Induction base: Consider any node B such that the edge $S \rightarrow B$ is in $\mathcal{G}$. From the translation, $< S : \{\} \longrightarrow B > \in \mathcal{G}$. $\mathcal{G}$ permits the edge $S \rightarrow B$ and hence $S <_S B$ in $\mathcal{G}$.

The corresponding justification in the ITMS is valid, since there are no negative children of B. So, $S \in L^{\Pi}(B)$, hence, B is IN and $S <_I B$. So, the lemma holds for the nodes that are immediately next to the focus in topological order.

Induction hypothesis: Consider any other node B. By induction hypothesis, for all $A <_T B$, the lemma is true, i. e.

$C <_I A \Leftrightarrow C <_S A$ in $\mathcal{G}$, and

$L_{\Pi}(A) \neq OUT \Leftrightarrow$ some path $\pi(s.\tau.A)$ is permitted by $\mathcal{G}$.

Induction step for proof of (1)

($\Rightarrow$):Let $C <_I B$, i. e. $C \in L^{\Pi}(B)$. Then there is a valid justification $< A : \{X\} \longrightarrow B >$, such that $C \in L^{\Pi}(A) \cup \{A\}$. So, we have a link $A \rightarrow B$ in $\mathcal{G}$. The validity of the justification implies A is IN. By induction hypothesis, there is a valid path $\pi(S.\sigma.A)$ from focus to the node A. We observe that C may be A itself or $C \in L^{\Pi}(A)$ (equivalently,$C <_I A$). In either case, the valid path to A has C on it. Now, from the validity of the justification (see the definition), it follows that the path $\pi(S.\sigma.A \rightarrow B)$ is valid in $\mathcal{G}$. Since the node C is on the path, $C <_S B$ in $\mathcal{G}$.

($\Leftarrow$):Let $C <_S B$. Then there is a path $\delta = \pi(S.\tau.A \rightarrow B)$ permitted by $\mathcal{G}$. By induction hypothesis, $C <_S A$ i. e. $C \in L^{\Pi}(A)$. The justification $< A : \{neg(B)\} \longrightarrow B >$ is valid; it follows from the validity of δ. Hence, from closedness $L^{\Pi}(A) \cup \{A\} \subseteq L^{\Pi}(B)$. So $C \in L^{\Pi}(B)$ or equivalently, $C <_I B$.

Induction step for proof of (2).

The argument is similar to the proof of 1.

□

Now the notion of *overrides* in ITMS and *preempts* in the net can be established formally .

Corollary 2 A is overridden for B by C iff every path $\pi(s.\sigma.A \rightarrow B)$ (resp. $\pi(s.\sigma.A \not\rightarrow B)$) is preempted in $\mathcal{G}$ by some path $\pi(s.\tau.C \not\rightarrow B)$ (resp. $\pi(s.\tau.C \rightarrow B)$).

Proof : A→B(respectively, A↛B) and C↛B(respectively, C→B) are in $\mathcal{G}$. Then, A is overridden for B by C iff C $<_I$A. By the lemma 3, C $<_S$ A in $\mathcal{G}$ by some path $\pi(s.\tau.C.\tau_1.A)$. Hence, any path $\pi(s.\sigma.A \rightarrow B)$ is preempted by the path $\pi(s.\tau.C \not\rightarrow B)$ (respectively, by $\pi(s.\tau.C \rightarrow B)$).

The reverse can be proved similarly.i

□

The definition of closedness gives us a method to compute the labels of a node from the labels of the children. Groundedness imposes the condition that we do not bring in any node other than those in the labels of the children.

Lemma 4 If $< A_i : \{X_i\} \longrightarrow Y_i >$B, $1 \leq i \leq n$, be all the valid justifications for B, then

$$\bigcup\{L^{\Pi}(A_i) \cup \{A_i\}\} = L^{\Pi}(B)$$

Proof : (By induction).

For the focus node s , $L^{\Pi}(s) = \{\}$, hence the lemma is trivially true.

For any other node B , let $< A_i : \{pos(B)\} \longrightarrow neg(B) >$B be a valid justification for B. From closedness, we have $L^{\Pi}(A_i) \cup \{A_i\} \subseteq L^{\Pi}(B)$. Hence, for all such justifications, we have

$$\bigcup\{L^{\Pi}(A_i) \cup \{A_i\}\} \subseteq L^{\Pi}(B)$$

Suppose, the valid justifications are $< A_i : \{pos(B)\} \longrightarrow neg(B) >$B for some values of i . By groundedness, for an element p $\in$ B, there is a valid justification $< A_i : \{pos(B)\} \longrightarrow neg(B) >$B such that p $\in L^{\Pi}(A_i) \cup \{A_i\}$, so p $\in \bigcup\{L^{\Pi}(A_i) \cup \{A_i\}\}$. Hence we have

$$L^{\Pi}(B) \subseteq \bigcup\{L^{\Pi}(A_i) \cup \{A_i\}\}$$

From the above two subset relations we get the desired equality.

□

The above lemma suggests that we can build exactly one label(one set of nodes) for a node, from the labels of its children in an ITMS with a fixed preference set. The following result shows that given a preference set, we can label the nodes of ITMS in exactly one way. This follows because the choice at ambiguous nodes is uniquely determined by the preference set.

Corollary 3 There is a unique closed and grounded labeling for the ITMS $\Gamma = (N, \Sigma, \Pi)$.

Proof : (By induction)

For the focus node $L^{\Pi}(s) = \{\}$.

Consider any other node B. By lemma 4 above, $L^{\Pi}(B)$ is computed from the labels of antecedent nodes of B. By induction hypothesis, the labels of the antecedents are unique. Hence the label of B is also unique.

□

We now show that the PTMS and ITMS translation of a net are equivalent so far as the internal nodes are concerned. This allows us to use the simpler ITMS instead of the PTMS for finding the paths permitted in the nets.

Theorem 2 Let $\mathcal{G}$ be an inheritance net. Let the PTMS and ITMS translation of the net be (N_p, S_p, Π) and (N_i, S_i, Π) respectively with some preference set Π. Let the PTMS labeling be L_p^{Π} and the ITMS labeling L_i^{Π} Then, for every node $B \in N_i$, $L_p^{\Pi}(B) = IN$ iff $L_i^{\Pi}(B) = IN$.

Proof : Follows from lemmas 1 and 3.

□

Theorem 2 allows us to work with the ITMS instead of the more complex PTMS simultaneously ensuring that we do not lose any power by doing so. There is only one problem. ITMS does not distinguish between nodes with unresolved conflict and nodes for which there is no path from the focus, when there is no choice in the preference set for the conflict. Of course, this does not affect the relevant information about which proofs are permitted by the net. One may trace this loss of information to the absence of any special mechanism (like the conflict nodes of type cntB in the PTMS) in the ITMS. But since our ultimate goal is to compute skeptical inheritance from the intersection of credulous extensions, we always use cautious preference sets– sets having a choice at every ambiguity as we have discussed in the previous section. Hence the problem stated above never arises.

3.5 Skeptical Inheritance

Our chief objective in the present chapter is to compute skeptical inheritance in defeasible acyclic nets. Since the skeptical closure of the net is just the intersection of all the credulous extensions, one immediate way of solving the problem is by generating all the credulous extensions and then taking the intersection. This is what we did in the example 1.4. But, the number of such extensions may be large (exponential in the number of ambiguous nodes) and the computation may be very expensive. Hence, we want to avoid the computation of full extensions. This is done as follows.

We have discussed how a minimal and cautious preference set effectively represents an extension. Then, if a node, say A, holds in an extension, we can state this fact by

associating the corresponding preference set with A. Since a node may hold in more than one extension, a set of preference sets can be associated with the node. This set, called the *environment* of A represents the extensions in which the node holds. Now, if the environment of A represents all possible credulous extensions, then it means the node is in all the extensions and hence in the skeptical closure of the net.

Hence, to compute skeptical closure of an inheritance net $\mathcal{G}$, we have to do the following:

1. For each node in $\mathcal{G}$ compute the environment, and
2. Check if the environment represents all the extensions.

The main advantage of the procedure is computational efficiency, since now we avoid the computation of all credulous extensions.

For computational efficiency, these two steps must be carried out efficiently. We handle each step in the subsequent sections.

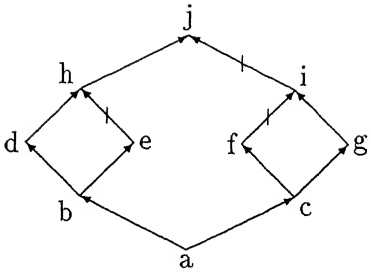
3.5.1 Computation of environments

The computation of environments in the acyclic net is done bottom-up. The focus node has a default environment, which says it holds in all preference sets. Then, we visit every node, say B , in a topological order and compute the environment of B from the already computed environments of its children .

The permission of a node (i. e. the node holds or not) depends only on the permission of its children and the specificity relation among them. Because of acyclicity the permission depends only on the nodes that are topologically earlier. Hence, only those ambiguities that are at nodes topologically earlier than B , can play a role for the permission of A. The set of nodes corresponding to these ambiguities is called a *relevant point set* for B .The other ambiguities are clearly redundant for A. Hence, when we choose the possible preference sets under which B may hold, we consider only those preference sets that have a positive or negative choice at all and only the nodes in the relevant point set only. These preference sets are called *relevant preference sets* for the node B . If P is a relevant point set, the set of all the relevant preference sets is denoted as $\mathcal{C}(P)$. It is defined as follows:

$$\mathcal{C}(P) = \{S \mid \forall n \in P \text{ either } n \text{ or } \neg n \in S\}.$$

One can do the following to compute the environment of the node B. Take $\mathcal{C}(.)$ of the relevant point set. This gives all possible relevant preference sets for B. One can then



Relevent Preference sets at j:
{ {h, i}, {h, ¬i}, {¬h, i} and {¬h, ¬i}
ν-set(j) = { {h, i}, {h, ¬i} }

Figure 3.5: Relevant preference sets.

check the permission relation for B under each relevant preference set. The environment consists of only those preference sets under which B holds . There may be conflict at B under a preference set. Then the choice at B is incorporated into the preference set under consideration. Now B holds in the preference set. This new preference set is then in the environment. In figure 2.8, suppose we want to find the environment of the node g. The set of all relevant preference sets is {{}, {e}, {¬ e}}. The node g holds under {¬e}. Under the other two preference sets there is a conflict at g , hence the new preference sets {g} and {e, g} are put in the environment of g.

$\mathcal{C}(.)$ generates a large number of relevant preference sets exponential in the number of nodes in the relevant point set), but actually most of the preference sets thus generated may be useless. In figure 3.5, the necessary condition for the node j to hold is that its positive child h should hold. But h holds only under the choice "h" since there is a conflict at h. So, j can hold only under those relevant preference sets in which "h" is. The set of relevant preference sets then is { {h, i}, {h, ¬i}}, half the size of the original set of preference sets.

To generate this smaller set, we take $\mathcal{C}(P)$ of a set of nodes P, where the nodes occur in the environments of the negative children of B, the node for which the environment is being computed. The preference sets of the positive children are extended by the sets in $\mathcal{C}(P)$. Care must be taken because the extended sets may contain both positive and negative choices for a node. We ignore such sets. The final set is called the ν-set of B . In figure 3.5, we can compute the ν-set of j . The result is the same as is given the previous paragraph.

We define a ν-set as follows.

Definition 23 Let $\mathcal{A}^+$ be the union of the environments of the positive children of B, i. e. $\mathcal{A}^+ = \bigcup \mathcal{E}(A_i)$, where A_i is a positive child of B .

Let P^- is the set of nodes that occur in the environments of the negative children of B .

Algorithm ComputeEnvironment;
Let $\mathcal{G}$ be an acyclic defeasible net with the focus S .

1. Environment(S) = $\{\{\}\}$.
   For all other nodes A , Environment(A) = $\{\}$
2. For each node B in topological order do:
   Let ν be the ν-set of [$\neg$] B .
   $\forall E \in \nu$ do:
   if [$\neg$]B holds under E then $E \in$ Environment([$\neg$] B)
   else if [$\neg$] B holds under $E \cup [\neg]B$ then $E \cup [\neg]B \in$ Environment([$\neg$] B).

Figure 3.6: Computation of environments.

Then, ν-set(B)= $\{E \cup P \mid E \in \mathcal{A}^+, P \in \mathcal{C}(P^-)$ and $E \cup P$ is a preference set$\}$

The definition of ν-set($\neg$B) is symmetrical. Here, we extend the preference sets of the negative children by choices at the nodes that occur in the environments of the positive children.

Since the focus has no children, the relevant point set for it is empty, hence the set of relevant preference sets for the focus is $\{\{\}\}$. The focus holds in any preference set, in particular in the empty preference set. Hence the environment of the focus is $\{\{\}\}$. An empty environment $\{\}$ denotes that the node does not hold in any extension. Now, we give an algorithm (refer figure 3.6) for computation of environments of nodes in the acyclic inheritance net. This basically encodes the ideas explained above.

Consider a particular node B. The restriction to the relevant preference sets filters out a number of useless preference sets. Since any node outside of the relevant point set is redundant for B, a relevant preference set may be taken to stand for all the naive preference sets that are its supersets. From the definition of ν-sets it is clear that for each preference set in the ν-set([$\neg$]B), at least one of the positive(respectively, negative) children holds and for all other preference sets none of the positive(resp. negative) children holds. Hence, restriction to ν-sets further filters out the preference sets in which none of the positive children hold and hence the node [$\neg$]B does not hold. Since, in effect, the algorithm in figure 3.6 considers all the the preference sets, it rightly computes the environment for the node B . From this discussion we have the following result.

Theorem 3 Algorithm ComputeEnvironment in figure 3.6 computes the environment of all the nodes in an acyclic defeasible net.

Proof : The proof follows from the discussion above .

□

We observe two things at this point.

- At step 2. (a) of the algorithm, we have to determine whether a node holds under a preference set. The simplistic solution is to compute the credulous extension for the preference set and then check for the node in the extension. This is clearly not desirable. We can avoid this computation by maintaining, with each relevant preference set in the environment of a node A, the set of nodes that are more specific to A . This set is nothing but the label of a node in an ITMS with the relevant preference set as the parameter. Then, from this information at the children of the node B , the node under consideration, we can know whether a node C overrides another for B. The costly computation of extensions is thus avoided.

- When we attach labels for each relevant preference set in the environment of a node, we can throw away some relevant preference sets that are redundant for the node.(Note that "relevance" is a necessary but not sufficient condition for redundancy. If a preference set is not relevant, either it is not in the environment or it is redundant. But even when it is relevant and is in the environment, it may be redundant, because there might be smaller preference sets with exactly the same proofs for the nodes). Since the label effectively captures all the valid proofs, we delete those preference sets that have subsets with the same label. In the expanded algorithm given later we use this filter(we call it "minimize") to make the environment smaller. Since only the redundant sets are deleted, and the smaller set effectively represents all the redundant supersets, the deletion does not result in any loss of information.

3.5.2 Checking environments for skeptical inheritance

The second phase in the computation of skeptical inheritance is to check whether the environment of a node represents all the credulous extensions. We do this as follows. The environment of a node is encoded as a formula in standard propositional logic.

We know that a cautious preference set stands for exactly one credulous extension. All the naive preference sets that are supersets of this cautious preference set also stand for the same extension(since the extra nodes are superfluous). The set of all cautious

preference sets exhaust all the choices at ambiguous nodes, and hence represent all the credulous extensions. So, to compute skeptical inheritance, one must check whether the environment of a node represents all the cautious preference sets. But, since we do not know which preference sets can be cautious without actually computing the credulous extensions, this method is unsuitable. However, there is a very simple observation that obviates this problem: the set of all cautious preference sets is equivalent to the set of all the naive preference sets, so far as representation of extensions is concerned. This is because the choices at ambiguities are exhausted by the cautious sets, and their supersets can exhaust all the choices at other nodes. Hence, we now check whether the environment of a node represents all the naive preference sets. Since we know all the naive preference sets, the checking is feasible now. We have the following lemma.

Lemma 5 A node holds in all extensions iff it holds in all the naive preference sets.

Proof :

($\Leftarrow$): Suppose a node holds in all the naive preference sets. If it does not hold in an extension, it does not hold in a corresponding cautious preference set P. P can be extended to a naive set W by any arbitrary (positive or negative) addition of nodes in $N \setminus P$ to P . The node does not hold in W, a contradiction.

($\Rightarrow$): Suppose a node does not hold in a naive set W . Since W defines an extension, the node does not hold in that extension.

□

Since the environment of a node consists of only relevant preference sets, the node holds under any preference set that is a superset of any of the preference set in the environment(the extra nodes are superfluous). The node does not hold under the rest of the preference sets, because the environment has already exhausted all possible preference sets at the time when it is built. Hence we have the following result.

Lemma 6 A node A holds in a preference set E iff $\exists P \in \mathcal{E}(A) \ni P \subseteq E$.

Proof :

($\Leftarrow$): If $E \supseteq P \ni P \in \mathcal{E}(A)$ then the nodes in $E \setminus P$ are redundant since P is a relevant preference set. Since A holds under P, it also holds under E.

($\Rightarrow$): Let A holds under E . Either E is relevant or it is not. If it is relevant it is in $\mathcal{E}(A)$. If it is not, then some nodes in it are redundant. So, a subset of E is in $\mathcal{E}(A)$.

□

If a subset of a preference set P is in an environment $\mathcal{E}$, then we write it as $\mathcal{E} \vdash P$. From the preceding lemma, then, we have: A node B holds under any preference set P *iff* $\mathcal{E}(B) \vdash P$.

The preceding lemma implies that the environment of a node represents a set of naive preference sets, since each preference set in the environment can be extended to many naive preference sets by putting in either positive or negative choices at redundant nodes. The problem is to determine whether all the $2^{|N|}$ naive sets are represented by the environment. We do this by encoding the environment as a formula in standard propositional logic and checking its validity. This connection is easily established because of the identical nature of naive preference sets and boolean semantics of propositional logic. We describe it below.

Let Λ_{N+} be a propositional logic with the propositional symbols from the set of positive nodes in N and $\top$ as the top element (interpreted as *true*) and $\bot$ as the bottom element (interpreted as *false*). An interpretation in Λ_{N+} is a set $\{\text{either } P \text{ or } \neg P \mid P \in N\}$.

Now, define a transformation $\mathcal{T}$ on the environments as follows.

$$\mathcal{T}(\mathcal{E}) = \begin{cases} \bot & \text{if } \mathcal{E} \text{ is empty} \\ (\bigwedge P_1) \bigvee (\bigwedge P_2) \bigvee \ldots & \text{if } \mathcal{E} = \{P_1, P_2, \ldots\} \end{cases}$$

$\bigwedge P_i$ is $\top$ if P_i is empty else it is the conjunction of the nodes in the preference set P_i.

A naive preference set is equivalent to an interpretation in the logic. This is rather obvious, because the definitions match.

Lemma 7 For a set I , where I is an interpretation in Λ_{N+}, $I \models \mathcal{T}(\mathcal{E})$ iff $\mathcal{E} \vdash I$.

Proof :

$I \models \mathcal{T}(\mathcal{E}) \iff I \models \bigwedge P_i$ for some $P_i \iff I \supseteq P_i \iff \mathcal{E} \vdash I$

□

We have the main result from the above lemmas.

Theorem 4 A node holds in all the extensions iff $\mathcal{T}(\mathcal{E}_{min}(A))$ is satisfiable in all the interpretations of Λ_{N+}, or equivalently, iff $\neg(\mathcal{T}(\mathcal{E}_{min}(A)))$ is unsatisfiable in Λ_{N+}.

Proof :

A node holds in all the extensions iff it holds in all the naive sets, i. e. for each naive set I , $\mathcal{E}(A) \vdash I$. By lemma 7, this is so iff $\mathcal{T}(\mathcal{E})$ is satisfiable by all the naive sets. Since the set of naive preference sets and the interpretations match, the theorem follows.

□

By this theorem, we can check if a node is in the skeptical closure of a net from its environment. The unsatisfiability of the propositional formula can be checked by some theorem proving technique such as resolution.

3.6 The Algorithm

Using the results in the previous sections, we present an algorithm to compute inheritance information in an acyclic defeasible net. All the inheritance information is computed with respect to a given node, the focus.

3.6.1 Design of the algorithm

The algorithm expands the basic algorithm ComputeEnvironment. In order to make the check of validity easier, we associate with each node a set of pairs $< E, L >$ where E is a preference set and L is the set of nodes more specific than the node–called label of the node. Like the environment, the labels are built up as we go bottom-up in topological order in the net.

The set of EL-pairs is called EL(n), for the node n. Then the environment of the node Env(n) can be given as: $\{E_i \mid < E_i, L_i > \in EL(n)\}$. EL(.) has the pairs for all preference sets P, for which $L^P(n) \neq OUT$. The extra advantage comes in when we want to check whether a node holds under an environment. Then we may want to check if a node B is overridden for another node A under a given preference set P. With EL(.) we can do it easily: find out a pair $< E, L > \in EL(B) \ni E \subset P$ and $A \in L$. Such a pair is present in EL(B) if and only if B is overridden by A. Thus we need not compute the full credulous extension. The EL-pairs at the child nodes provide sufficient information to determine if the node holds or not under an environment.

We now present the algorithm in figure 3.7. The main subroutine it calls is in figure 3.8.

Theorem 5 The algorithm ExpandedComputeEnvironment computes the environment of all nodes in the net $\mathcal{G}$.

Proof :

The algorithm ExpandedComputeEnvironment in figure 3.7 just fills the details of the earlier algorithm ComputeEnvironment. Hence, from theorem 3, the result follows.

algorithm **Algorithm** ExpandedComputeEnvironment;
Input $\mathcal{G}$; /* the inheritance net */
s; /* the focus node */
Output $\mathcal{E}_{min}$ of each node in $\mathcal{G}$;
Begin
Sort the nodes of $\mathcal{G}$ topologically;
Let the topological order be $(n_1, n_2, \ldots, n_k)$;
EL(s) =< {}, {{}} >;
/* focus node holds always */
$\forall n_j \neq s, 1 \leq j \leq k$, EL($n_j$) = {} and EL($\overline{n_j}$) ={};
/* initially no other node holds */
/* consider only nodes that are topologically later */
/* than s. */
For j = i+1 to k
$\mathcal{E}_{min}(n_j)$ = Compute-Env-From-Children(n_j
End;/* ComputeEnvironment */

Figure 3.7: Algorithm for computing minimal environments.

□

We conclude the chapter with a few examples.

Example 3.3 Consider the network in figure 1.6. One topological order is c, r, e and g. Let c be the focus, EL(c) is $\{< \{\}, \{\} >\}$. Since the only child of r is c and c is positive, EL(r) is $\{< \{\}, \{c\} >\}$. Now, EL(e) is constructed from only the positive children c and r, hence EL(e) is $\{< \{\}, \{c, r\} >\}$. Consider node g. It has a positive child e and a negative child r. Since the environments of both the children are $\{\{\}\}$ ν-set(g) is $\{\{\}\}$. Both r and e hold under the preference set $\{\}$ and e is preempted for g by r since label of e contains "r". Hence, EL(g) is $\{\}$, which means g does not hold. We can see that EL($\neg g$) is $\{< \{\}, \{c, r\} >\}$.

Example 3.4 Consider the network in figure 2.9. We discuss only the interesting points. Let a be the focus. Then the environments of nodes a, b and d are $\{< \{\}, \{\} >\}$, $\{< \{\}, \{a\} >\}$ and $\{< \{\}, \{a, b\} >\}$ respectively.

Node e has a positive child b and a negative child c. ν-set of e is $\{\{\}\}$, and e does not hold under $\{\}$ since there is a conflict. However, it holds under {e}, and we have EL(e) = $\{< \{e\}, \{a, b\} >\}$. Since g has a sole positive child e, EL(g) is $\{< \{e\}, \{a, b, c\} >\}$.

Algorithm Compute-Env-from-Children;
Input A node n_j;
Output Minimal Environment of n_j–$\mathcal{E}_{min}(n_j)$
Begin

Let $X = \{n \mid n \in pos(n_j), EL(n) \neq \{\}\}$
Let $Y = \{n \mid n \in neg(n_j), EL(n) \neq \{\}\}$
/* since the nodes in X and Y are earlier in the
topological order, we know the EL(.) of each
of the nodes in X and Y */
/* no valid children */
If $X = \{\} and Y = \{\}$ **Then Break;**
/* only valid negative children */
Else If X = {} **Then** EL($\neg n_j$) = $\bigcup$EL'(n_l), $n_l \in$Y.
where EL'(n_l) = $\{< E, L' > | < E, L > \in \text{EL}(n_l)$ and $L' = L \cup \{n_l\}$
/* only valid positive children */
Else If Y = {} **Then** EL'(n_j) = $\bigcup$ EL(n_j), $n_l \in$X.
where EL'(n_l) = $\{< E, L' > | < E, L > \in \text{EL}(n_l)$ and $L' = L \cup \{n_l\}$
/* valid positive and negative children */
Else
$\nu(n_j) = \{< E, L > |$ E $\in \nu$-set(n_j) and $\exists < E', L > \in \text{EL}(n_l), n_l \in$ X $\ni E' \subseteq$ E$\}$
$\nu(\neg n_j) = \{< E, L > |$ E $\in \nu$-set($\neg n_j$) and $\exists < E', L > \in \text{EL}(\neg n_l), n_l \in$ X $\ni E' \subseteq$ E$\}$
For each $< E, L > \in \nu(n_l)$ **do Begin**
X' = $\{n_k \mid n_l \in$ X and Holds(n, E)=true$\}$
Y' = $\{n_k \mid n_l \in$ Y and Holds(n, E)=true$\}$
For each $n_l \in$ X' **do Begin**
If $\forall n \in$ Y', either $n \notin$ Y' or PreEmpted(n, X')= true
Then $< E, L \cup \{n_l\} > \in$ EL(n_j),
Else $< E \cup \{n_j\}, L \cup \{n_l\} > \in$EL($n_j$);
end;
For each $n_l \in$ Y' **do Begin**
If $\forall n \in$ X', either $n \notin$ X' or PreEmpted(n, Y') = true
Then $< E, L \cup \{n_l\} > \in$EL($\overline{n_j}$),
Else $< E \cup \{n_j\}, L \cup \{n_l\} > \in$EL($\overline{n_j}$);
end;
End;
End;
return(Env(minimize(EL(n_j)));
End/*ComputeEnvironment*/

Env(n) = { E |< E, L>$\in$ EL(n) }
minimize(EL) deletes from EL the pairs $< E_1, L >$ such that there is a pair $< E_2, L >$ and $E_2 \subset E_1$. This is the third filter discussed previously.
Holds(n, E) = true iff $\exists E' \in Env(n) \ni E' \subset E$.
PreEmpted(n, Z) = true iff $\exists m \in Z \ni m \in L(n)$.

Figure 3.8: Subroutine for computation of environment of a node from its children.

Now, ν-set of f is { {e}, {¬ e}}. Node f holds under {¬e}, since the negative child e does not hold then. There is a conflict under the environment {e}, so we can put {e, f} in the environment of f. So, EL(f)= {<{e, f}, $\{a, b, d\}$ >,< {¬e},$\{a, b, d\}$ >}.

Node h has no negative children. We ignore the labels and see that the environment of h is { {e}, {¬e}, {e, f} }. The propositional transformation of this set is valid. Hence, h is in the skeptical closure.

4 Cyclic and Mixed Nets

In this chapter, we extend the ideas of the previous chapter for the more general case of cyclic and mixed nets. The basic intuition we follow for these cases is different from the existing notions.

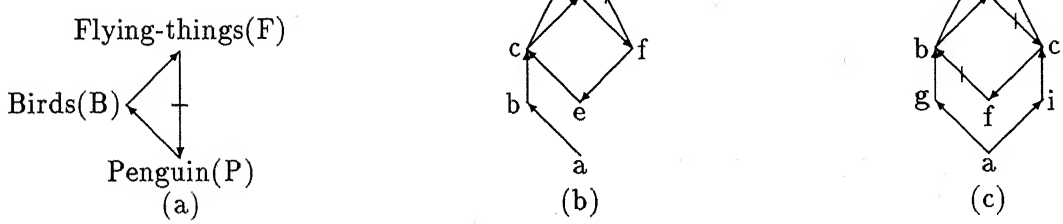
4.1 Cyclic Nets

In Chapter 2, we have pointed out the difficulty associated with cyclic nets. In this section we explore the cause of the problem. For easier reading, we give the figures here again.

Consider figure 4.1. (a). If we apply the ITMS-translation for acyclic nets here with a premise justification $< : \{\} \longrightarrow P >$, we get $\neg P$ IN, which is clearly absurd since A has been assumed to be IN through the premise justification in the first place.

To overcome this problem. one may suggest a modified definition of validity of a justification $< A : \{X\} \longrightarrow [\neg]B >$ by adding an extra condition *L(¬B)=OUT(L(B)=OUT)* to the set of conditions already mentioned. But then this is the condition for credulous extensions(depending on which justification's validity is first checked by the TMS) and will undo the effect of the parameter in ITMS

See figure 4.1. (b). The traditional point of view [1] is to consider neither b nor d more



specific than the other. So f is ambiguous with respect to a.

In both the cases above, we can see a pattern in the traditional viewpoint: *Derive A is IN, derive B is IN using A, derive [¬]A is IN using B.* In figure 4.1.(a), we have P IN *a priori*(through a premise justification). It is used to derive that F is IN; then F is used to derive that ¬A is IN. In figure 4.1.(b), we have b IN, which is used to derive that d is IN, which in turn is used to derive that b is IN. We call this the problem of *Cyclic Proofs.* A proof which is not cyclic is a *Normal Proof* or an *Acyclic proof.*

We may immediately suggest a remedy to cyclic proofs: *a proof of A can be derived using B only if there is a proof of B which is not already derived from a proof of A.*

A difficulty arises when nodes have more than one proof(derivational sequences); some cyclic and some normal. One must then be able to distinguish between them. For example, see figure 4.1.(c). Allowed proofs of f can be one of the sequences $a \rightarrow b \rightarrow d$, $a \rightarrow b \rightarrow d \rightarrow e \rightarrow g \rightarrow f$ and $a \rightarrow c \rightarrow g \rightarrow f$. While the first two proofs can not be used for a proof of d, the third one can be used to do so. This necessitates the recording of different paths(denoting different proof sequences) with each node and not just the more specific nodes as in the acyclic case. One may note, however, that recording of paths is a generalization since we may get the set of more specific nodes from the union of the nodes on the paths.

4.2 A Theory for Cyclic Nets

As for acyclic nets, we translate the cyclic nets into a set of justifications in a CITMS (Cyclic Inheritance TMS).

Formally, A CITMS is a triple $\Gamma =< N, \Sigma, \Pi >$ with the usual meanings. But the labeling is different here. The IN label for a node is replaced by a set of sets of nodes. If a node N has a label $X = \{p_1, \ldots, p_n\}$, where p_i are sets of nodes, it means

1. the node N is IN
2. each p_i is a valid path upto N, and
3. each node in p_i is more specific than N(with respect to the focus node)

With the change in labeling the definition of specificity changes as follows.

Definition 24 A node A is more specific to a node B in the CITMS–written $A <_C B$–iff $\exists \delta \in L^{\Pi}(B) \ni A \in \delta$ and $\nexists \sigma \in L^{\Pi}(A) \ni B \in \sigma$.

The definition of overriding then becomes:

Definition 25 A node A is *overridden* for B by a node C, where C is a child of B with opposite polarity to that of A, iff $B <_C A$.

Example 4.1 See figure 4.1.(b). Assume Π to be empty. L^{Π}(b)={{a}}, L^{Π}(d)={{a b c}}. Hence B is overridden by d in {d}).

The definitions of validity, closedness and groundedness are changed suitably as follows.

Let $\Gamma =< N, \Sigma, P >$ be a CITMS. Let $L^{\Pi} : N \rightarrow \mathcal{P}(\mathcal{P}(N)) \cup \{OUT\}$ be a labeling under P.

Definition 26 A justification $\sigma \in \Sigma$ is *valid* for $[\neg]B$ in L^{Π} iff

1. either B is the focus, i. e. $\sigma = < : \{\} \longrightarrow [\neg]B >$, or
2. A child of B, say A, is IN, there is an acyclic path from the focus to B through A, no other child of B overrides A for B and there is no conflict, i. e. $\sigma = < A : \{X\} \longrightarrow [\neg]B >$, and $L^{\Pi}(A) = IN$, $\exists \delta \in L^{\Pi}(A) \ni B \notin \delta$, $\forall N \in X \ni L^{\Pi}(N) = IN$, A is not overridden for B by N and N is overridden for B by A, or
3. there is a conflict at B and "B" is chosen, i. e. $\sigma = < A : \{X\} \longrightarrow [\neg]B >$, and $L^{\Pi}(A) = IN$, $\exists \delta \in L^{\Pi}(A) \ni B \notin \delta$, $\exists N \in X \ni L^{\Pi}(N) = IN \ni \exists \delta_1 \in L^{\Pi}(N)$ where $B \notin \delta_1$, A is not overridden for B by N and N is not overridden for B by A and $[\neg]$"B" $\in \Pi$

When a justification is valid, there are acyclic paths from the focus to the consequent node, say B, through its children. The definition of closedness now assigns all those labels to B that stand for these acyclic paths.

Definition 27 A labeling L^{Π} is *closed* for the CITMS iff

1. if $< : \{\} \longrightarrow >$A $\in \Sigma$ is valid in L^{Π} then $\{\} \in L^{\Pi}(A)$ (denoting $L^{\Pi}(A) \neq OUT$).
2. if $< A : \{X\} \longrightarrow B > \in \Sigma$ is valid in L^{Π} then $\forall \delta \in L^{\Pi}(A) \ni B \notin \delta, \delta \cup \{A\} \in L^{\Pi}(B)$.

Since the labeling and the definition of validity breaks the cyclicity of proofs, the definition of groundedness follows easily. Groundedness of labeling imposes that only the labels denoting acyclic proofs can be assigned to a node.

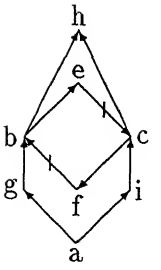


Figure 4.1: Cyclic dependency in inheritance nets.

Definition 28 A labeling L^{Π} for the CITMS as *grounded* iff there is a linear ordering $\{(A_1, Y_1), (A_2, Y_2), \ldots\}$ of the set $\{(A, Y) \mid A \in N, L^{\Pi}(A) = Y \neq OUT\}$, such that for every pair (A_i, Y_i)

if $Y_i = \{\{\}\}$ then $< : \{\} \longrightarrow A_i > \in \Sigma$ else

$\forall \beta \in Y_i \exists \sigma = < A_h : \{X\} \longrightarrow Y > A_i$ such that

$h < I, \sigma$ is valid in $L^{\Pi}, \exists \delta \in Y_h \ni A_i \notin \delta$ and $\beta = \delta \cup \{A_h\}$.

We consider only closed and grounded labelings. Here, the definition of closedness and groundedness ensures that all and only the valid acyclic proofs are labeled IN(neq OUT).

4.3 Skeptical Inheritance in Cyclic Nets

As in the acyclic case, one needs to compute the preference sets for which a node is IN. In the acyclic nets, the environment(set of all such preference sets) of a node is computed once and for all from the environment of its children. But in the cyclic nets, a direct computation is not possible because the environment of a child may also depend upon the environment of the parent node. This cyclic dependency is illustrated in figure 4.1.

Example 4.2 When the nodes a, b and c are IN, the environment of d is $\{\{\}\}$, since f is OUT. But actually, as one can see, environment of d is $\{\{\mathrm{d}\}\{\neg \mathrm{g}\}\}$.

Because of the problem stated above, in order to compute environments, we have to go over the cycles. The following subsection shows that the computation can be done in a finite number of steps.

4.3.1 Computation of Minimal Environments

The nodes of the CITMS are ordered in a sequence in some arbitrary manner, such that we can refer to a node by its position in the sequence. Let the focus node be the 0-th

node.

Let Ψ^t be an n-tuple $< \mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_n >$, where each $\mathcal{E}_i$ denotes the minimal environment of the i-th node, say A_i, at stage t . Ψ^0–the initial tuple, at stage 0–is $< \{\{\}\}, \{\}, \{\}, \ldots, \{\} >$.

Let $\lambda(\Psi^t)$ be defined as $< \lambda(\mathcal{E}_1), \lambda(\mathcal{E}_2), \ldots, \lambda(\mathcal{E}_n) >$, where $\lambda(\mathcal{E}_i$ is the environment computed from the $\mathcal{E}_j$'s of the children n_j's of the node n_i.

Let $\widehat{\Psi^t}$ be a fixed point of λ over the argument t , such that $\lambda(\widehat{\Psi^t}) = \widehat{\Psi^t}$.

We observe the following:

1. The environment computed at a node at each stage is a superset of the environment at the previous stage, because once a preference set is included in the environment, it remains in it in all subsequent stages. Since the set of naive preference sets is the upper limit for any environment, each environment saturates at some stage. Hence, the fixed point exists.

2. Since the labeling is unique for a given preference set, we know for certain whether a node holds in that preference set or not. Hence, we know for certain whether the preference set is to be included in the environment or not. Thus, the environment at a node at each stage is unique. Hence the fixed point is also unique.

The computation of minimal environments for the nodes of the net is now conceptually easy. We illustrate this through an example below. Note that in the case of acyclic nets, this iterative computation of the fixed point is trivial and corresponds exactly to the computation done in topological order. Once we get the minimal environments, we can compute skeptical inheritance using the technique of translating the environments into propositional formulas and checking there validity(satisfiability in all the interpretations).

Example 4.3 See figure 4.2. It gives the environments of the nodes (other than the focus) in the net shown in the figure 4.1 at various stages. The environment of the focus a is $\{\{\}\}$ always. The last set of environments is the fixed point for λ. The environment of h in this fixed point shows that this node is in the skeptical closure. From the discussion in Chapter 2, we know that this result is correct.

t	$\mathcal{E}$(()b)	$\mathcal{E}$(()c)	$\mathcal{E}$(()d)	$\mathcal{E}$(()e)	$\mathcal{E}$(()f)	$\mathcal{E}$(()g)	$\mathcal{E}$(()h)
0	{}	{}	{}	{}	{}	{}	{}
1	{{}}	{{}}	{}	{}	{}	{}	{}
2	{{}}	{{}}	{{}}	{}	{}	{{}}	{}
3	{{}}	{{}}	{{}}	{{}}	{{}}	{{}}	{}
4	{{}}	{{}}	{{d}}	{{}}	{{}}	{{g}}	{}
5	{{}}	{{}}	{{d}}	{{d}}	{{g}}	{{g}}	{{d}{g}}
6	{{}}	{{}}	{{d}{¬g}}	{{d}}	{{g}}	{{g}{¬d}}	{{d}{g}}
7	{{}}	{{}}	{{d}{¬g}}	{{d}{¬g}}	{{g}{¬d}}	{{g}{¬d}}	{{g}{¬ d} {d}{¬ g}}
8	{{}}	{{}}	{{d}{¬g}}	{{d}{¬g}}	{{g}{¬d}}	{{g}{¬d}}	{{g}{¬ d} {d}{¬ g}}

Figure 4.2: Computation of environments of nodes in cyclic nets.

1. $a \rightarrow \ldots \rightarrow b \Rightarrow c \Rightarrow d \Rightarrow e$
2. $a \rightarrow \ldots \rightarrow b \rightarrow c \not\rightarrow d \Leftarrow e \Leftarrow f$
3. $a \rightarrow \ldots \rightarrow b \rightarrow c \not\Leftrightarrow d \Leftarrow e \Leftarrow f$

Figure 4.3: Positive and negative strict consequences.

4.4 An ITMS for Mixed Nets

4.4.1 Strict Consequences

The presence of strict links/paths in the net imposes certain conditions on the inferences allowed in the net. For example, see figure 4.3.

In (1), if we know that $a \rightsquigarrow b$ is allowed then the strict links decree that $a \rightsquigarrow c$, $a \rightsquigarrow d$ and $a \rightsquigarrow e$ are also allowed. In (2), if $a \not\rightsquigarrow d$ is allowed, from contraposition of strict links we must also have $a \not\rightsquigarrow e$ and $a \not\rightsquigarrow f$. Similarly in (3), when $a \rightsquigarrow c$ is allowed, we also have $a \not\rightsquigarrow d$,$a \not\rightsquigarrow e$ and $a \not\rightsquigarrow f$. These strict consequences can be divided into positive or negative.

Definition 29 Positive strict consequences of a node x are defined as

$$\kappa_{\mathcal{G}}(x) = \{y \mid \mathcal{G} \text{ contains a strict positive path from x to y}\}$$

Negative strict consequences of a node x are defined as

$$\overline{\kappa_{\mathcal{G}}}(x) = \{y \mid \mathcal{G} \text{ contains a strict negative path from x to y}\}$$

4.4.2 The ITMS

The requirement of the ITMS are the following.

1. It should represent the strict links in some way.

2. It must be able to take care of the problems of indirect conflicts, new preemption relations and cycles in the net. An immediate observation is that when negative strict links($\not\Leftrightarrow$) are present, cycles are introduced for the negation in both ways.

The first requirement is fulfilled rather easily. A strict link $A \Rightarrow B$ is represented by an ITMS justification $< A : \{\} \longrightarrow > B$. To allow contraposition, with each such strict link we introduce a justification $< \neg B : \{\} \longrightarrow > \neg A$.

The other justifications are of the form $< : \{\} \longrightarrow [\neg] > N$(premise justification) and $< A : \{X\} \longrightarrow [\neg] Y > B$. The labeling of nodes is as in cyclic nets(set of paths). The definitions of validity, closedness and groundedness are also same. Only differences are in the definitions of preference sets and the translation of the net to ITMS.

The change in definitions is due to the presence of strict consequences.

Definition 30 A *preference set* $P \subseteq N \cup \overline{N}$ is such that there are no x $\in P$ and $\overline{y} \in P$, where $y \in \kappa_{\mathcal{G}}(x)$ and $z \in P$ where $z \in \overline{\kappa_{\mathcal{G}}}(x)$.

Also because of the presence of strict consequences, the notion of children in the net is also extended.

Definition 31 The set of positive children of a node, *pos(n)* is defined as
pos(n)= $\{x \mid y \rightarrow x \in \mathcal{G}\} \cup \{x \mid \text{n} \in \kappa_{\mathcal{G}}(x)\} \cup \{y \mid y \rightarrow x \text{ and n} \in \kappa_{\mathcal{G}}(x)\}$.

The set of negative children of a node, *neg(n)* is defined as
neg(n)= $\{x \mid y \not\rightarrow x \in \mathcal{G}\} \cup \{x \mid \text{n} \in \overline{\kappa_{\mathcal{G}}}(x)\} \cup \{y \mid y \rightarrow x \text{ and n} \in \overline{\kappa_{\mathcal{G}}}(x)\} \cup \{y \mid y \not\rightarrow x \in \mathcal{G} \text{ and x} \in \kappa_{\mathcal{G}}(n)\}$.

Example 4.4 See figure 4.4. The positive children of n are a, b and c and d. The negative children are g, h, i, j, l and m.

Definition 32 [Translation]

Let $\mathcal{G}$ be the inheritance network with nodes M. the ITMS translation of $\mathcal{G}$ is the ITMS network $\Gamma = (N, \Sigma, P)$, where,

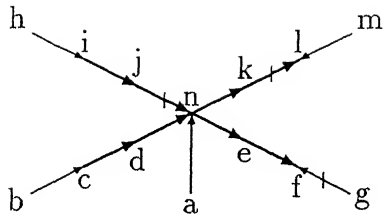
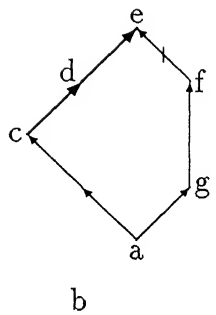


Figure 4.4: Mixed net with postive and negative strict consequences.



$< : \{\} \longrightarrow a >$
$< a : \{\} \longrightarrow b >$
$< a : \{\} \longrightarrow g >$
$< g : \{\} \longrightarrow f >$
$< b : \{f\} \longrightarrow c >$
$< f : \{b, c, d\} \longrightarrow \neg e >$
$< c \longrightarrow d >$
$< d \longrightarrow e >$
$< \neg d \longrightarrow \neg c >$
$< \neg e \longrightarrow \neg d >$

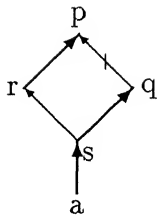
Figure 4.5: Net showing indirect ambiguity.

$N = M \cup \{\neg B \mid A \not\rightarrow B \in \mathcal{G}$ or $A \not\Leftrightarrow B \in \mathcal{G}\}$,
P is a parameter(the preference set), and
Σ is the smallest set of justifications such that

1. if $A \Longrightarrow B \in \mathcal{G}$ then $< A : \{\} \longrightarrow > B$, $< \neg B : \{\} \longrightarrow > \neg A \in \Sigma$.
2. if $A \not\Leftrightarrow B \in \mathcal{G}$ then $< A : \{\} \longrightarrow > \neg B$, $< B : \{\} \longrightarrow > \neg A \in \Sigma$.
3. if $A \longrightarrow B \in \mathcal{G}$ then $< A : \{X\} \longrightarrow Y > B \in \Sigma$, where X =pos(B) and Y =neg(B).
4. if $A \not\rightarrow B \in \mathcal{G}$ then $< A : \{X\} \longrightarrow Y > \neg B \in \Sigma$, where X =neg(B) and Y =pos(B).

When there are no strict links in the net, these changed definitions coincide with those in the defeasible nets.

Example 4.5 Figures 4.6 and 4.5 show two mixed nets and the corresponding ITMS's. The IN nodes for different preference sets has been given. They match with the results of [6].



$$< : \{\} \longrightarrow A >$$
$$< A \longrightarrow B >$$
$$< \neg B \longrightarrow \neg A >$$
$$< B \longrightarrow D >$$
$$< \neg D \longrightarrow \neg B >$$
$$< C \longrightarrow E >$$
$$< \neg E \longrightarrow \neg C >$$
$$< B : \{D\} \longrightarrow C >$$
$$< D : \{B, C\} \longrightarrow E >$$

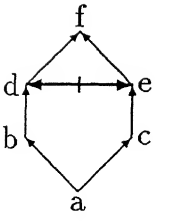
Figure 4.6: Net showing new preemption relation.

4.4.3 Computation of Environments

Since the nets may contain cycles, we will have to iterate over the cycles till all the preference sets are computed. We carry over the method of the previous section on cyclic nets with the extended notion of positive and negative children to compute the minimal environments of the nodes. In case of defeasible nets, these definitions of "children' reduces to the simple definitions, hence the defeasible nets are specific cases of this general formalism.

Thus, we can now compute the environment of nodes and hence compute skeptical inheritance in the most general type of nets. We conclude the chapter with an example.

Example 4.6 See the mixed net in figure 4.7. If one has the interpretation: a = Nixon, b = quaker, e = republican, d = pacifist, e = non-pacifist and f = lover of football, then we can say that irrespective of whether Nixon is a pacifist or not, he is a football-lover. So, foot-balleer should be in the skeptical closure. The table iin figure 4.8 shows the environments of the nodesat various stages of computation. After the 8'th stage, the environments stabilise. At ths point, we see that the environment of the node f is {{D}, {E}, {¬E}, {¬D}}. The propositional transformation of this environment is valid, hence f is in the skeptical closure of the net.



(a). A mixed net.

Figure 4.7: Strict net where f is in the skeptical closure.

t	$\mathcal{E}(a)$	$\mathcal{E}(b)$	$\mathcal{E}(c)$	$\mathcal{E}(d)$	$\mathcal{E}(e)$	$\mathcal{E}(f)$
0	{{}}	{}	{}	{}	{}	{}
1	{{}}	{{}}	{{}}	{}	{}	{}
2	{{}}	{{}}	{{}}	{d}	{e}	{}
3	{{}}	{{}}	{{}}	{{d}{¬e}}	{{e}{¬d}}	{{d}{e}}
4	{{}}	{{}}	{{}}	{{d}{¬e}}	{{e}{¬d}}	{{¬d}{d}{e}{¬e}}
5	{{}}	{{}}	{{}}	{{d}{¬e}}	{{e}{¬d}}	{{¬d}{d}{e}{¬e}}

Figure 4.8: Skeptical inheritance in strict nets.

5 Conclusion

In this thesis, we have given a new inference technique for nets representing multiple inheritance with exception. It employs two strategies for resolution of conflicts among proofs, namely *preemption* and *choice*. We first deal with acyclic defeasible nets. In order to provide a semantics for the nets and for the inference, we translate the nets into a modified TMS, called PTMS(Parametrized TMS). The modification is motivated by the need to incorporate the "choice" at ambiguous nodes . Since PTMS justifications for the inheritance nets are large in number and the interconnection among them is involved, we suggest a more natural and simple TMS, called ITMS(Inheritance TMS), which exploits the regularity of information in the nets to generate a small number of justifications. The validity of ITMS justifications match closely with the definition of validity of proofs in the nets. We claim that because of its simplicity and directness of representation, ITMS can represent more general type of inheritance information which can not be represented in the existing nets.

Then, we give a procedure for computing skeptical inheritance in acyclic defeasible nets. The idea is to compute all the sets of choices—called preference sets—under which the nodes hold and then to show that these sets effectively account for all the preference sets. We avoid computing all the credulous extensions to compute skeptical inheritance.

We extend the above ideas to the general case of cyclic nets. We discuss the need for change in the existing definitions of the specificity relation in these nets and give a procedure to translate the nets into an ITMS with a different labeling scheme. The cyclic interdependence among proofs necessitate a more general method of computing skeptical inheritance. The extension hence to the most general case of mixed nets (i. e. nets containing strict links) is done easily by a change in the definition of "children" in the nets which takes care of problems arising from the strict consequence of proofs in mixed

nets.

Translating proofs in nets to proofs in a PTMS (or, to the equivalent ITMS) has two consequences:

- The inference mechanism with a preference set is captured by the PTMS, which is actually a TMS. Since the correspondence between TMS and AEL(Autoepistemic Logic) is already established in the literature, we get the semantics of the nets and the new inference procedure in terms of AEL. The association of preference sets with the nodes basically states that the nodes are labeled IN in some PTMS's. Hence, from the connection between TMS and AEL, we get a semantics for skeptical inheritance too.

- The new method of computation of skeptical inheritance is a generalization of a procedure given by Stein. The procedure by Stein dealt with only acyclic defeasible nets with a very restrictive specificity relation, which led to a preemption strategy, called *on-path* preemption. We claim that in real-life domains (which are what the inheritance nets are expected to represent) the proposed method will have better performance than the naive way of computing skeptical inheritance: that of computing all the credulous extensions and then taking their intersection.

In the following few sections, we justify some decisions taken in the thesis.

5.1 Why not a direct translation to AEL ?

One of the major goals of the thesis is to show that computing skeptical inheritance is actually a subproblem of more general commonsense reasoning. Since AEL models commonsense reasoning, in order to capture the inheritance problem in the framework of AEL we have to show that the inheritance relations translate into a set of AEL sentences and that the translation is both sound and complete with respect to the notion of inference in the nets. We have adopted an indirect approach. We translate the nets into a TMS. The lemma 1 establishes the soundness and completeness through the equivalence between the nodes that hold in the net and the nodes that are labeled IN in the PTMS. The equivalence between TMS and AEL is known. Hence, effectively we put the inheritance problem in the framework of AEL. Why do we adopt this indirect approach?

The main reason is that the computation of semantics in AEL is not easy. The semantics of a set of statements in AEL is given by sets having certain properties. These

properties are self-referential (AEL models beliefs of a rational agent who may have beliefs about his beliefs, beliefs about beliefs about beliefs and so on). Hence, these sets are fixed points of the properties. The point to note is that computing these fixed points may not be easy. So, checking soundness and completeness w. r. t inheritance nets would lead us to study the properties of such sets, i. e. existence, uniqueness etc. under different conditions such as when the inheritance relations form an acyclic net. On the other hand, the proofs are very simple in the TMS. The closedness and groundedness criteria are simple, intuitive and sufficient to compute the IN nodes (the current belief set). The TMS-labeling algorithms are well-understood. So one can use the TMS itself to compute the nodes that hold in a net under certain preference sets. Since the hard work of showing equivalence between TMS and AEL has already been tackled in the literature, the indirect approach has been preferred.

5.2 Why ITMS ?

The main reason for introducing the ITMS is that it significantly reduces the number of justifications without any loss of information. A defeasible link in the net is translated into exactly one justification in the ITMS; a strict link into two justifications, the extra one corresponds to the contraposition. The reduction in number of justifications is a major advantage, since the PTMS can give rise to a very large number of justifications for even small nets [1].

The other advantage is that the ITMS justifications correspond directly to the inheritance relations and a check for validity of a justification has a natural correspondence with the bottom-up construction of proofs in the inheritance net The justifications allow more general inheritance relations to be represented without any difficulty, for example, relations of type non-mammals are oviparous (egg-laying) can be represented by allowing the antecedent to be negative nodes as well . The inheritance nets in the existing literature are unable to represent such relations.

5.3 Complexity Issues

Perhaps the point we have harped the most is the computational efficiency of skeptical inheritance. We have relinquished the "naive" idea of computing all the credulous extensions and taking the intersection in favour of computing the environments and checking

the satisfiability of a corresponding propositional formula. But, as yet, we have not shown, why the first idea is "naive" and how the proposed idea fares better. Let us check the complexity of the two notions.

In the first case, there can be exponential number of extensions resulting from all the choices at the ambiguities. Hence, to compute skeptical inheritance, we need at least exponential time (in the number of ambiguous nodes).

In the worst case analysis the new idea does not give any improvement over the order of complexity: constructing environments requires finding the ν-set, which is exponential in the number of ambiguities below the negative children and the checking phase is actually a satisfiability problem, hence it also requires at least exponential time. Where is the advantages then?

In real-life domains, the inheritance nets are expected to contain a large amount of membership information. Hence, computation of a single extension may also prove to be expensive, because of the sheer volume of data. Computing all the credulous extensions, even if the number is few, is out of question. Our proposal behaves much better in this case. We do not expect to have many ambiguities: most of the ambiguities may have been resolved by special domain-specific conflict resolution strategies. Therefore the the preference sets and ν-sets will be small. Thus the computation of ν-sets and checking of environments can be easy in spite of the apparent exponential complexity of the problem. So we can compute skeptical inheritance without replicating computations. In the "naive" approach, this seems difficult to avoid.

5.4 Future work

Future work can address itself to the following:

- The notion of resolution of conflicts has been based on two strategies: *specificity* and *preference* in the thesis. One may follow many other strategies for such resolution such as *evidences*[13] or *ordering the paths based on probabilities* assigned to them. But the basic technique of resolution "first try to resolve the conflict through the domain specific strategy, in case of failure choose" remains the same. The resolution strategy may be domain dependent and it would be worthwhile to experiment with different strategies. This would entail changing the criteria by which node labels are assigned. For example, in the thesis specificity was the criterion, based on which validity of proofs in the nets (equivalently, validity of justifications in the ITMS)

was checked and labels were assigned.

- When a new relation is discovered among objects/classes, we may have to add links to the net. There is as yet no incremental method for computing the inheritance information and it has to be computed afresh whenever a new link is added.

- The other generalization is by introduction of inheritance of n-ary relations along with the unary membership relation. This will help us in reasoning with a more enriched language in the inheritance nets. The following example is from [13].

  1. Generally, engineering students love Maths.
  2. Metallurgy students hate discrete maths.
  3. Metallurgy students are engineering students, and discrete maths courses are maths course.
  4. Calculus I is a maths course, while Set Theory is a discrete maths course.
  5. John is an engineering student, while James is a metallurgy student.

  From the above facts, we can conjecture that John loves Calculus I, while James hates Set Theory.

These avenues await further attention and fuller investigation.

Bibliography

[1] Gerhard Brewka. A simple tms-based theory of inheritance. Preprint. Submitted to Proc. IJCAI-91.

[2] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[3] D. W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *AAAI-83*, pages 104–108, 1983.

[4] B. Haugh. Tractable theories of multiple defeasible inheritance in ordinary nonmonotonic logics. In *AAAI-88*, pages 421–426, 1988.

[5] John F. Horty, Richmond H. Thomason, and David S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. Technical Report CMU-CS-87-175, Computer Science department, CMU, October 1987.

[6] John F. Horty, Richmond H. Thomason, and David S. Touretzky. Mixing strict and defeasible inheritance. In *AAAI-88*, pages 427–432, 1988.

[7] Kurt Konolige. Hierarchic autoepistemic theories for non-monotonic reasoning: Preliminary report. In M. Reinfrank, J. de Kleer, M. L. ginsberg, and E. Sandewall, editors, *LNAI 346*, pages 42–59. Springer-Verlag, 1988. Proceedings of the second International Workshop on Non-monotonic Reasoning.

[8] Kurt Konolige. On the relation beween default and autoepistemic logic. *Artificial Intelligence*, 35:343–382, 1988.

[9] D. Makinson and K. Schlechta. Floating conclusions and zombie paths: two deep difficulties in the directly skeptical approach to defeasible inheritance nets. *Artificial Intelligence*, 48(2):199–210, March 1991. (Research Note).

[10] John Mcarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[11] Drew McDermott and Jon Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[12] R. Moore. Semantical considerations in nonmonotonic logic. *Artificial Intelligence*, 13:75–94, 1985.

[13] T. K. Prasad. The semantics of inheritance networks. Technical Report WSU-CS-90-04, Computer Science department, SUNY at Stony Brook, February 1990.

[14] Michael Reinfrank, Oskar Dressler, and Gerhard Brewka. On the relation between truth maintenance and autoepistemic logic. Preprint, submitted to the Proceedings IJCAI-89.

[15] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[16] Erik Sandwell. Non-monotonic inference rules for multiple inheritance with exceptions. In *Proceedings of the IEEE*, volume 74, pages 1345–1353, 1986.

[17] L. A. Stein. Skeptical inheritance: Computing the intersection of credulous extensions. In *IJCAI-89*, pages 1153–1159, Detroit, MI, 1989.

[18] David S. Touretzky. *The mathematics of Inheritance Systems.* Morgan Kaufmann Publishers, 1986.

[19] David S. Touretzky, John F. Horty, and Richmond H. thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *IJCAI-87*, pages 476–482, 1987.